ABC INFORMATIQUE

N° 21

COURS D'INFORMATIQUE PRATIQUE ET FAMILIALE

Traitement de texte

Les progiciels

Le portable Osborne I

Figures sur Commodore 64

EDITIONS ATLAS

M 6062 - 21 - 12 F 85 FB - 3,80 FS - $ 1.95

aBc INFORMATIQUE
cours d'informatique pratique et familiale
EDITIONS ATLAS

Édité par ÉDITIONS ATLAS s.a., tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Belgique : ÉDITIONS ATLEN s.a., Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, Montréal Nord.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, Mezzovico.

Réalisé par EDENA s.a., 29, boulevard Edgar-Quinet, 75014 Paris. Tél. : 320-15-01.

Direction éditoriale : J.-Fr. Gautier. *Service technique et artistique* : F. Givone et J.-Cl. Bernar. *Iconographie* : J. Pierre. *Correction* : B. Noël.
Publicité : Anne Cayla. Tél. : 202-09-80.

VENTE

Les numeros parus peuvent être obtenus chez les marchands de journaux ou, à défaut, chez les éditeurs, au prix en vigueur au moment de la commande. Ils resteront en principe disponibles pendant six mois après la parution du dernier fascicule de la série. (Pour toute commande par lettre, joindre à votre courrier le règlement, majoré de 10 % de frais de port.)

Pour la France, s'adresser à ÉDITIONS ATLAS, tour Maine-Montparnasse, 33, avenue du Maine, 75755 Paris *Cedex 15*. Tél. : 538-52-70.

Pour les autres pays, s'adresser aux éditeurs indiqués ci-dessous.

SOUSCRIPTION

Les lecteurs désirant souscrire à l'ensemble de cet ouvrage peuvent s'adresser à :

France : DIFFUSION ATLAS, 3, rue de la Taye, 28110 Lucé. Tél. : (37) 35-40-23.

Belgique : ÉDITIONS ATLEN s.a., 55, avenue Huart-Hamoir, 1030 Bruxelles. Tél. : (02) 242-39-00. Banque Bruxelles-Lambert, compte n° 310-0018465-24 Bruxelles.

Canada : ÉDITIONS ATLAS CANADA Ltée, 11450 boulevard Albert-Hudon, Montréal Nord, H 1G 3J9.

Suisse : FINABUCH s.a., ÉDITIONS TRANSALPINES, zona industriale 6849 Mezzovico-Lugano. Tél. : (091) 95-27-44.

RELIEZ VOS FASCICULES

Des reliures mobiles permettant de relier 12 fascicules sont en vente chez votre marchand de journaux.

ATTENTION : ces reliures, présentées sans numérotation, sont valables indifféremment pour tous les volumes de votre collection. Vous les numéroterez vous-même à l'aide du décalque qui est fourni (avec les instructions nécessaires) dans chaque reliure.

En vente tous les vendredis. Volume II, n° 21.

ABC INFORMATIQUE est réalisé avec la collaboration de Trystan Mordrel *(secrétariat de rédaction)*, Jean-Pierre Bourcier *(coordination)*, Patrick Bazin, Jean-Paul Mourlon, Claire Rémy *(traduction)*, Ghislaine Goullier *(fabrication)*, Marie-Claire Jacquet *(iconographie)*, Patrick Boman *(correction)*.
Crédit photographique, couverture : DGT.

Directeur de la publication : Paul Bernabeu. Imprimé en Italie par I.G.D.A., Officine Grafiche, Novara. Distribution en France : N.M.P.P. Tax. Dépôt légal : mai 1984. 25845. Dépôt légal en Belgique : D/84/2783/27.

A NOS LECTEURS

En achetant chaque semaine votre fascicule chez le même marchand de journaux, vous serez certain d'être immédiatement servi, en nous facilitant la précision de la distribution. Nous vous en remercions d'avance.

Les Éditions Atlas

Esprit d'enfance
Les enfants sont plus réceptifs aux nouvelles technologies que bien des adultes, toujours réticents à l'idée de devoir apprendre à nouveau. La versatilité du microprocesseur permet de supprimer toute limite d'âge inférieure pour les jouets électroniques et les ensembles éducatifs.
(Cl. Tony Sleep / Hamleys.)

Jeux d'enfants

Les derniers « jouets » éducatifs ont autant de puissance de traitement que votre ordinateur et font usage de techniques de programmation analogues.

Les microprocesseurs n'appartiennent pas qu'aux ordinateurs; on en trouve désormais dans de nombreux appareils électroménagers (machines à coudre ou à laver), mais aussi dans certaines... serrures. Les fabricants de jouets y ont eu recours également, en particulier pour le contrôle des trains électriques et des modèles réduits de voitures. Texas Instruments, un des grands de l'informatique, a su se créer un marché dans le domaine du jeu éducatif, en utilisant les mêmes méthodes. Son premier essai fut une petite calculatrice, qui posait des problèmes d'arithmétique très simples. The Little Professor (« le Petit Professeur ») connut un immense succès qui se prolongea bien après la sortie de Speak & Maths (« Paroles et maths »), pourtant plus sophistiqué.

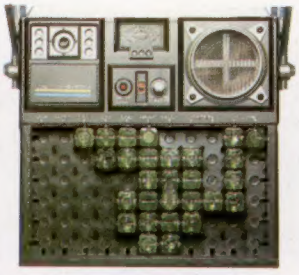
Ces deux appareils disposaient de la « puce parole » de Texas Instruments, dont était équipé l'ordinateur familial TI-99/4A, retiré du marché à la fin de 1983 après que la firme eut subi de lourdes pertes financières. Speak & Spell (en France : « la Dictée magique »), lancé en 1978, a un vocabulaire de plusieurs centaines de mots. L'ensemble dispose d'un clavier alphabétique (et de quelques touches supplémentaires) à membrane assez semblable à celui du ZX-81 de Sinclair. Quand il appuie sur une touche en bas du clavier, l'enfant s'entend demander « à voix haute » d'entrer un mot. Chaque lettre frappée est affichée sur un écran à cristaux liquides; lorsque le mot est complet, l'appareil annonce, de la même façon, s'il est correctement écrit ou non.

Speak & Maths fonctionne de façon analogue, mais pose des problèmes arithmétiques. De par leur succès, ces deux jouets ont permis à Texas Instruments de s'emparer d'une large part du marché et d'aborder un domaine nouveau, très différent de ceux qu'il avait occupés jusque-là.

Un troisième jouet parlant de TI, Touch and Tell (« le Livre magique »), est sans doute plus récréatif qu'éducatif. Il fait usage de feuilles de plastique sur lesquelles sont imprimés des chiffres ou des images qu'un codage magnétique permet d'identifier. Quand l'enfant touche l'un de ces motifs, l'appareil dit de quoi il s'agit.

La synthèse de la parole est la plus complexe de toutes les techniques informatiques utilisées par les fabricants; elle est très employée dans les versions domestiques des jeux d'arcades les plus populaires. Le microprocesseur s'est également implanté dans un autre secteur du marché du jouet, celui des véhicules autoguidés. Big Trak est le plus connu d'entre eux : on peut le programmer en entrant des instructions à l'aide d'un clavier situé sur le haut. Il ressemble à une tortue, et peut être raccordé à un ordinateur, grâce à son port parallèle.

Parmi les autres dispositifs fonctionnant à l'aide d'un microprocesseur, on citera : le « Jeu de Simon », dans lequel l'enfant doit reproduire une séquence de notes musicales et de lumières clignotantes; Playskool's Maximus, qui lui ressemble un peu, mais peut aussi poser des problèmes d'arithmétique, à la façon du Little Professor; et de nombreux robots. Pour les enfants plus grands (et les adultes!), une variété de systèmes sont proposés en kit et attendent pour être montés : des composants électroniques encapsulés sont installés sur un boîtier, permettant ainsi la réalisation d'appareils très simples.



Electroni-Kit
Comme son nom l'indique, Electroni-kit est un ensemble à monter soi-même qui permet de réaliser plusieurs appareils électroniques : transistors, amplificateurs...
Les composants sont encapsulés dans du plastique transparent et montés sur un boîtier, à l'aide des schémas de montage fournis par le constructeur. Le kit le plus sophistiqué du lot rassemble les éléments de base d'un micro-ordinateur rudimentaire, dont le but est d'enseigner certaines opérations très simples.
(Cl. Ian McKinnell.)

Maximus
C'est là un produit plus sophistiqué dû à MB Electronics. Il ressemble à une calculette, mais c'est en fait une variante du Jeu de Simon plus élaborée : il faut reproduire une série de notes de musique, d'images, de rimes, de formes, ou citer l'orthographe de certains mots.

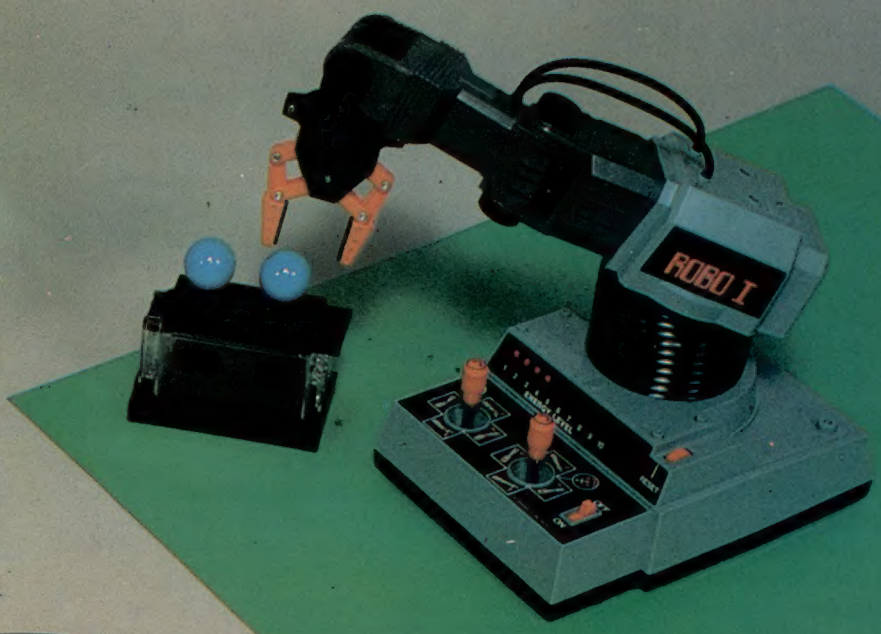
Milton Bradley

Texas Instruments
TI fit son apparition sur le marché du jeu éducatif vers 1975, avec The Little Professor, un appareil de type calculatrice qui posait des problèmes d'arithmétique. Peu de temps après, la firme mit en vente la Dictée magique, qui enseigne l'orthographe et dispose de la synthèse de parole. Lorsqu'il presse une touche, l'enfant s'entend demander d'épeler un mot en le tapant au clavier. De telles techniques ont été reprises plus récemment pour des jeux mathématiques très simples et des histoires élémentaires destinées aux tout petits.

Simon
Ce produit de MB Electronics est une version sur microprocesseur du Jeu de Simon bien connu des enfants anglo-saxons (qui s'apparente à notre « Jacques a dit »). Il s'agit de reproduire une série de notes de musique (dont chacune est accompagnée d'une lumière clignotante) donnée par l'appareil en s'aidant des quatre quadrants de couleur situés sur le dessus.

Texas Instruments

Robo-1
Le Robo-1 produit par Tomy est un bras articulé conventionnel guidé par l'intermédiaire de deux manches à balai. La plus grande surprise, c'est le prix — moins de dix fois celui du modèle le moins cher. Bien entendu, le Robo-1 est beaucoup moins solide : il est fait de plastique moulé sous injection et non de métal. Il ne comporte pas de moteurs pas à pas et c'est en fait l'utilisateur qui surveille du regard son fonctionnement et fait les corrections nécessaires.



Tomy Robo-1

Big Trak
Il ressemble à ces véhicules destinés aux jeunes enfants, mais Big Trak est en fait un robot d'appartement. Complètement autonome, il est pourvu d'un clavier installé sur le dessus, grâce auquel on peut programmer certaines indications de direction et de distance. On peut le connecter à un ordinateur familial (par l'intermédiaire d'un port série ou parallèle), après quelques manipulations. Le véhicule peut alors être soumis à un programme qui règle son déplacement, et peut prévoir telle ou telle réaction suivant les circonstances.

Milton Bradley

Ian McKinnell

Orthographe

Les traitements de texte avec vérification de l'orthographe ne manquent pas sur le marché; mais le style et la grammaire sont aujourd'hui visés.

La création d'ordinateurs capables de comprendre les langues naturelles est encore loin. Un des objectifs des ordinateurs de la cinquième génération, qui devraient apparaître vers les années 1990, est d'aboutir à la traduction du français en japonais par exemple. Les machines de traduction entre langues sont étudiées depuis quelques années. Il en existe déjà pour des textes simples usant d'un vocabulaire restreint comme dans les rapports. Il faut pourtant systématiquement reprendre les résultats pour les corriger. Les difficultés de la traduction sont telles qu'un texte traduit par ordinateur dans une langue et retraduit dans la langue d'origine par un autre ordinateur se trouve être dénaturé. Le problème est celui de la compréhension du contexte qui donne un sens différent au même mot employé dans deux phrases différentes. L'emploi des mots intervient aussi pour nuancer l'expression. Un verbe utilisé comme adverbe ou comme adjectif n'a pas la même connotation. Le sens n'est pas univoque aux termes, et seul l'usage d'une langue le laisse pressentir. L'ordinateur est donc bien en peine de pouvoir cataloguer, répertorier les langues.

Il faudrait que les ordinateurs soient dotés d'énormément de mémoire pour stocker les différents sens possibles; mais il faudrait aussi qu'ils soient capables d'intelligence pour choisir parmi toutes les nuances. C'est le domaine de la recherche en intelligence artificielle. Recherche qui n'est pas encore très avancée mais qui suscite des dépenses de plus en plus grandes.

Parlons plutôt de la différence entre la syntaxe et la sémantique. La syntaxe recouvre l'ensemble des règles de construction de la langue. Elle peut être facilement assimilée par l'ordinateur, car elle est limitée et bien déterminée. Les ordinateurs individuels recourent à une syntaxe qui leur est propre (témoins les messages d'erreurs de syntaxe). La sémantique, elle, renvoie à l'étude du sens ou plutôt des sens possibles.

Dans les années cinquante, Noam Chomsky posa les bases d'une théorie moderne de la langue et de la grammaire. Bien qu'il ne fût pas particulièrement préoccupé par l'informatique, les retombées de ses travaux ont directement contribué au développement des interpréteurs et des compilateurs pour l'écriture de programmes. La traduction par ordinateur a profité également des mêmes principes. L'idée d'une grammaire universelle est pour beaucoup à l'origine des langages de programmation.

L'écriture par ordinateur passe par des programmes de traitement de texte. Il s'agit de faciliter la saisie, l'archivage et l'impression de texte. Mais ces programmes facilitent également l'ordonnancement d'un texte, donc sa présentation, et l'orthographe est vérifiée automatiquement. La vérification grammaticale et stylistique est également prévue. Bien que ces programmes n'apportent rien de nouveau au problème de l'intelligence artificielle, ils sont néanmoins intéressants. Leur organisation, leur programmation, leur présentation méritent d'être abordées.

Les programmes qui vérifient l'orthographe d'un texte comportent un dictionnaire stocké sur disque, dont le répertoire couvre de 25 000 à 50 000 mots. Ces progiciels vous permettent également d'intervenir sur le lexique, en ajoutant vos propres termes (termes techniques ou références). Le problème de la place mémoire se pose, puisque 1 octet (ensemble de 8 bits) ne peut coder qu'un seul caractère alphanumérique, selon les codes ASCII. Ainsi, en comptant une moyenne de 5 caractères par mot, cela signifie que pour 30 000 mots, il est nécessaire de disposer de 150 octets de mémoire.

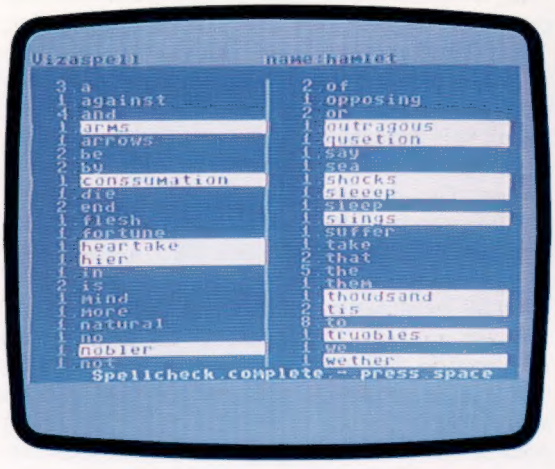
Cette capacité de stockage excède largement celle des ordinateurs domestiques. Il faut donc réduire cette masse de données. Deux techniques y pourvoient. D'abord, si l'on estime que le programme n'a besoin que de caractères minuscules (une routine fera la conversion), et que l'on peut éliminer aussi les chiffres et les caractères de ponctuation, on réduira considérablement les données.

Il est donc possible d'établir un dictionnaire à partir de 32 caractères de base, au lieu des 128 caractères ASCII (ou 256, si l'on inclut les caractères graphiques). Chaque mot pourrait être stocké sur 5 bits au lieu de 8. Ainsi le mot « ordinateur » tiendrait sur 50 bits. Les cinq premiers bits coderaient la lettre « c », les cinq suivants « o » (les trois derniers bits du premier octet, et les deux premiers bits du deuxième octet) et ainsi de suite. La deuxième technique, pour réduire la mémoire stockée, consiste à coder non plus une lettre, mais un ensemble de lettres revenant souvent sous la même forme. Ainsi, un octet pourrait représenter un groupe donné de caractères. Un drapeau signalerait que le codage renvoie non pas à un seul mais à un ensemble de caractères. Votre micro-ordinateur utilise certainement cette technique en stockant en mémoire vive les mots clés BASIC, PRINT ou NEXT par exemple, sur un seul octet chacun.

Pour un programme de vérification d'orthographe, cette technique de grouper les caractères fréquents est utilisée pour les débuts de mots. Les lettres formant des syllabes revenant fréquemment sont ainsi répertoriées. VIZA-SPELL et VIZA-WRITE sur le Commodore 64 utilisent les deux techniques pour stocker 30 000 mots sur seulement 65 K.

Le plus difficile pour un programme de vérification d'orthographe est de comparer les mots lus dans le texte avec le dictionnaire. Une recherche binaire pourrait être envisagée, mais elle prendrait beaucoup trop de temps. Pour un document de 1 millier de mots, cela prendrait des heures! Idéalement chaque terme devrait être vérifié à sa saisie, mais cela se révèle impossible en termes de programmation. Aussi, un document donné est-il vérifié dans son ensemble, soit sur disque, soit en mémoire vive pour les plus gros ordinateurs. Le programme parcourt le document et compile par ordre alphabétique les mots qu'il contient. Un texte important est souvent constitué à 50 % seulement d'une centaine de mots. Cette constatation permet également d'apprécier l'usage des mots dans le texte et de signaler les répétitions.

Un algorithme simple prend alors un par un les mots relevés dans le texte, et recherche dans le dictionnaire leur équivalent. Le temps d'une telle recherche est fortement réduit; il ne prend que quelques minutes, quatre dans le cas de VIZASPELL, et quelle que soit la longueur du document. Les mots non trouvés sont soit listés, soit signalés dans le texte même. Pour chaque mot



ainsi signalé, les options suivantes sont offertes à l'utilisateur :

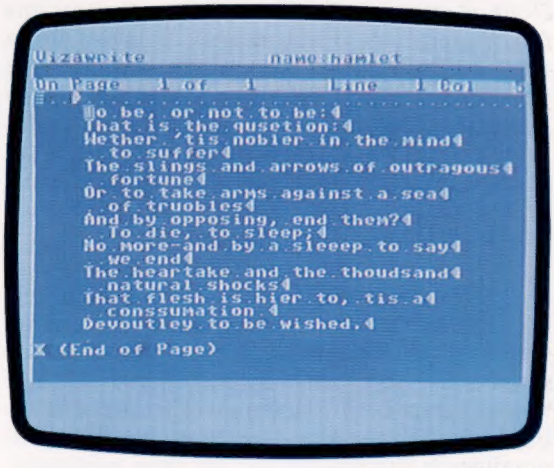
1) erreur d'orthographe ou de frappe; le mot doit être corrigé;

2) le mot est bien écrit et doit être ajouté au dictionnaire;

3) le mot est exact mais a peu de chances d'être réutilisé (par exemple, il fait partie d'une adresse). On ne doit pas s'en préoccuper et ne pas l'ajouter au dictionnaire.

Les programmes qui vérifient les constructions syntaxiques fonctionnent de la même manière. Ils vérifient par exemple la présence d'une lettre majuscule en début de mot et se contentent de vérifier un nombre limité de points, ce qui signifie qu'un grand nombre d'inexactitudes grammaticales ne sont pas prises en compte. Mais déjà des chercheurs tentent d'aller au-delà de tous ces principes techniques pour s'attaquer à la « qualité » même du texte. Mais il faut reconnaître que les programmes qui vérifient le style d'un document n'en sont qu'à leur début. Ils répertorient un certain nombre de constructions sous forme d'exemples servant à identifier une mauvaise syntaxe et les expressions incorrectes. Généralement, ils proposent de meilleures tournures pour les phrases concernées en se référant à leur dictionnaire. Ils relèvent également les répétitions d'expressions dans un même paragraphe, mais aussi les phrases trop longues, ou inélégantes.

Écrire un petit programme de vérification d'orthographe, de syntaxe ou de style peut être



Ian McKinnell

Être ou ne pas être
Imaginez les questions de littérature aux examens sans avoir de problèmes d'orthographe...
Ce programme, Viza Spell s'en charge. Il vérifie l'orthographe d'un texte tapé sur traitement de texte (ici, l'exemple du monologue de Hamlet). Les mots utilisés sont listés par ordre alphabétique, et leur fréquence est calculée. Ils sont ensuite confrontés au dictionnaire pour identification. Les mots dont l'orthographe est différente ne sont pas reconnus, et sont signalés dans le texte. Les mots qui n'existent pas dans le dictionnaire sont également signalés. L'utilisateur a la possibilité de les y inclure. A vous de corriger l'orthographe des mots mal tapés s'il s'agit de mots figurant dans le dictionnaire sous la bonne orthographe. (Cl. Popperfoto.)

un excellent exercice informatique. Même un programmeur peu expérimenté en BASIC peut le faire, pourvu qu'il maîtrise bien les fonctions de maniement de chaînes de caractères sur son ordinateur. De par la complexité croissante des logiciels, les programmes de traitement de texte, quant à eux, comprendront bientôt de tels programmes de contrôle. Et pourquoi pas, ce que tout programmeur rêve :
'COMMAND > GÉNÉRER ARTICLE, LONGUEUR 1200 MOTS, COMMENCER'.

Progiciels

Des programmes générant des programmes d'applications permettent aussi de construire des jeux.

Pinball Construction Set
Ce progiciel est une sorte de générateur d'applications de jeux. L'utilisateur donne la présentation et la logique du jeu par un menu d'objets et d'outils graphiques.
(Cl. Ian McKinnell.)

Nous avons vu dernièrement un programme qui, moyennant certaines spécifications de la part de l'utilisateur, générait un programme capable de réaliser l'application désirée. De tels générateurs de programmes existent pour la plupart des micro-ordinateurs de gestion; il en existe également pour les ordinateurs domestiques, mais ceux-ci doivent disposer d'au moins un lecteur de disquettes.

De leur côté, les programmes générateurs d'applications sont d'un usage plus courant. Mais les programmes qui en résultent, contrairement à ceux produits par un générateur de programme, ne sont pas autonomes. Ils ont besoin, pour fonctionner, du générateur d'applications. Envisageons la création d'un programme de facturation par les deux méthodes, afin de mieux mettre en valeur leurs différences.

Si nous utilisons un générateur de programme, il doit d'abord être chargé à partir de la disquette. Lorsque l'utilisateur a répondu à toutes les questions concernant les fichiers, les enregistrements, les zones, les relations mathématiques, les dessins d'écran et les états à imprimer, le générateur de programmes demandera l'insertion d'une disquette vierge. Le nouveau programme sera sauvegardé sur cette disquette. Ce type de sauvegarde permet de multiplier les copies du nouveau logiciel et d'en assurer l'utilisation à de nombreuses personnes.

Le générateur d'applications peut sembler à cet égard moins satisfaisant. Une fois les spécifications données, les routines créées sont stockées sur le disque même du générateur. Si le programme créé est recopié sur un autre disque, ce dernier aura néanmoins besoin du générateur pour fonctionner. La présence physique du générateur d'applications sera donc toujours nécessaire à l'application elle-même.

Les générateurs sont en outre protégés contre la copie, ce qui interdit finalement toute possibilité de diffusion pour le logiciel d'application créé, à moins d'envisager autant d'acquisitions de générateurs que de copies de l'application.

Un générateur d'applications est un programme très sophistiqué destiné à une utilisation générale. Lorsqu'on spécifie l'application, on ne fait qu'assigner des valeurs à un certain nombre de variables importantes appelées paramètres. Ces derniers contrôlent le déroulement du programme, la structure des données, les dessins d'écrans et ceux des états à imprimer. Lors de la sauvegarde de l'application, seule la liste des paramètres est stockée. Cette dernière, appelée module d'application, est la référence d'exécution de l'application pour le générateur.

Certains progiciels vont encore plus loin en vous permettant de spécifier votre application sous la forme d'un langage très évolué (semblable au pseudolangage que nous avons utilisé pour développer une routine dans le cours de programmation BASIC). Ce listage est interprété par le générateur. L'interpréteur BASIC (si le programme de « génération » a été écrit en BASIC) peut également l'interpréter, ce qui pose un problème intéressant de hiérarchie de programmes.

Il n'est pas rare que les auteurs d'applications ne soient pas à l'origine de la création et de la commercialisation des générateurs eux-mêmes. C'est le cas de D-BASE II, le plus célèbre des progiciels gestionnaires de données. Celui-ci peut même à son tour être considéré comme un générateur d'applications, dans la mesure où ses modules d'applications consistent en des chaînes de commandes très sophistiquées sur les données. Il est donc possible d'élaborer des modules d'applications très spécialisés (comptabilité boursière par exemple) sans avoir à écrire un programme. Des applications aussi particulières (et donc de diffusion très limitée) peuvent être développées en se concentrant sur la

matière même du programme (le sujet), et non pas sur l'écriture du code. La réalisation du logiciel en est plus facile et évite toute erreur (sur la manipulation des fichiers par exemple). Le fonctionnement, le déroulement du programme sont rendus très fiables par leur conception originale dans le cadre du générateur. La grande diffusion du générateur, pour des applications très diverses, assure une parfaite fiabilité aux programmes ainsi créés.

La principale différence entre un générateur de programmes et un générateur d'applications réside surtout dans leur degré d'abstraction. Un générateur de programmes crée artificiellement du code, probablement en BASIC. L'homme n'y intervient pas, aussi reste-t-il abstrait, peu efficace et de qualité inférieure au code produit par un générateur humain. Le générateur d'applications, par contre, est très proche de l'utilisateur, et est très marqué par l'intervention humaine. Il est concret et reflète la démarche de son utilisateur, le concepteur de l'application. Ce dernier est souvent un spécialiste du domaine d'application lui-même. On peut citer Silicon Office, générateur d'applications en micro-informatique de gestion qui génère du code machine. Les programmes créés comportent des procédures de détection d'erreurs opérateur, et un ensemble de menus à l'écran pour manœuvrer les logiciels.

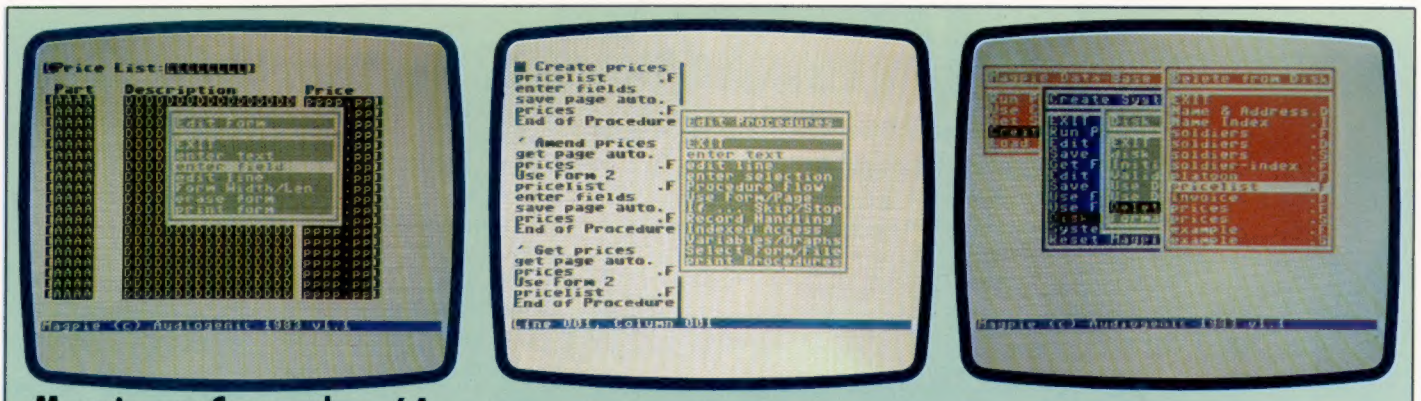
Toujours est-il que les générateurs d'applications ne sont pas limités à des programmes de gestion. Le meilleur exemple de progiciel qui sort du cadre des affaires est sans conteste le Pinball Construction Set dont le module d'applications est créé en spécifiant les données que demande la table du générateur.

Il faut distinguer en réalité le générateur de programmes de la programmation orientée vers les applications. Cette dernière consiste à permettre à l'utilisateur d'implémenter son application simplement, en spécifiant ses objectifs au programme. Même de simples programmes tableurs, destinés à des ordinateurs familiaux comme le Spectrum, par exemple, peuvent être considérés, à un certain degré, comme des générateurs d'applications, puisque l'on spécifie simplement les relations entre zones et que le progiciel se charge du programme.

Magpie, produit par Audiogenic pour le Commodore 64 avec un lecteur de disque, est un générateur d'applications orienté vers les affaires ou autres domaines « sérieux ». Il utilise largement la programmation fondée sur la définition d'objectifs à l'écran. Le type de relations entre les différents éléments des données est spécifié lors de la disposition du document à l'écran.

Ces principes s'appliquent à des degrés divers à un nombre de plus en plus grand de logiciels. Même si l'on ne peut pas considérer ces derniers comme de véritables générateurs de programmes. A leur première exécution, ces programmes conduits par paramètres poseront à l'utilisateur toute une série de questions et stockeront les réponses sur disque, avec le programme. Ces informations fourniront les détails nécessaires pour individualiser l'application. Un programme de comptabilité, par exemple, posera différentes questions relatives aux pratiques de gestion de l'entreprise qui l'utilisera : quels types de renseignements sont formulés dans les factures ou quelles sont, suivant les principales circonstances, les modalités de crédit qui sont autorisées? Autre exemple : un jeu de simulation demandera, avant de jouer, le niveau de difficulté recherché, c'est-à-dire le nombre de personnages, d'envahisseurs, de fusées, etc., et même, pourquoi pas, de choisir la nature des ennemis contre lesquels on souhaite se battre!

Les logiciels s'orientent de plus en plus vers l'utilisateur. Celui-ci est libéré de la programmation et peut intervenir très librement sur la destination du programme. Le logiciel va vers l'utilisateur au lieu du contraire, comme cela était le cas auparavant. Cette démarche est tout à fait remarquable dans la mesure où elle dispense de tâches fastidieuses.



Magpie sur Commodore 64

La première étape dans la création d'une application est de mettre au point le dessin des états qui seront produits (rapports et documents divers). L'utilisateur remplit les colonnes de lettres (A, D, P), et désigne ainsi les zones du gestionnaire de données à utiliser. Le traitement est ensuite spécifié par des instructions en langage de programmation évoluée que Magpie interprète. Voici les routines pour corriger les prix et les appeler depuis le disque (GET).

Magpie fonctionne par menus. Si une option est choisie (CREATE par exemple), une autre suivra après la présentation de CREATE. Cet écran montre le résultat de la succession suivante de choix : CREATE, puis DISK, puis DELETE, puis le fichier à supprimer, ici PRICE LIST.

Ian McKinnell

Sons incroyables

La commande ENVELOPE du BBC modèle B permet un son pratiquement sans limites.

Nous avons abordé précédemment la commande SOUND du BBC. Les capacités sonores du BBC seront pleinement utilisées lorsque SOUND se verra adjoindre la commande ENVELOPE. ENVELOPE permet à l'utilisateur de créer quatre sons imitant de manière satisfaisante les instruments conventionnels. En outre, les effets sonores permettent de reproduire des explosions ou des rafales d'armes de manière très convaincante!

La syntaxe de ENVELOPE est la suivante :

```
ENVELOPE N, T, PS1, PS2, PS3, NS1, NS2, NS3,
AR, DR, SR, RR, FAL, FDL
```

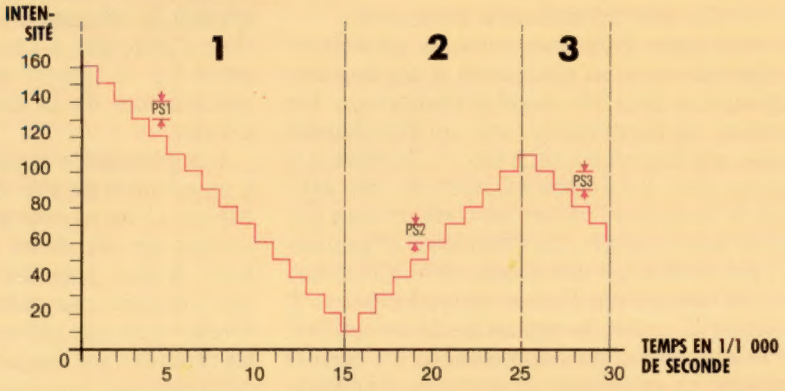
Le premier paramètre, N, donne le chiffre à attribuer à ENVELOPE, et permet d'identifier la valeur à associer aux commandes SOUND et SOUND &. Quatre valeurs sont possibles pour remplacer la valeur de VOLUME, affublée d'un nombre négatif (de 0 à − 15) par SOUND.

```
T (de 0 à 127) & (de 128 à 255)
```

Ainsi se présente le contrôle principal du temps. Il détermine la durée en centièmes de seconde de chaque palier d'élaboration de ENVELOPE. T = 5 signifie que chaque palier dans ENVELOPE dure 5 centièmes de seconde (0,05 seconde). En ajoutant 128 à la durée voulue pour un palier, le processus d'autorépétition de la gamme d'intensités (Pitch envelope) est supprimé. T = 5 + 128 = 133 donne une durée de 5 centièmes de seconde pour un palier dans une gamme d'intensités qui n'arrive qu'une seule fois pour une note.

L'utilisation de « pitch envelope/gamme d'intensités » peut prêter à confusion puisque ENVELOPE a été utilisé au préalable pour désigner un niveau sonore. Mais, ici, il s'agit de la variation d'intensité sur la durée d'une note. Cette notion est de peu d'utilité en musique, à moins de vouloir obtenir un « vibrato », mais elle est très précieuse pour les effets sonores militaires. Comme le diagramme l'indique, la gamme d'intensités est divisée en trois parties. Leur paramétrage se fait par l'intermédiaire des nombres PS et NS :

Gamme d'intensités



Relief de l'image

L'utilisation de figures sur Commodore 64 est attrayante.

Une des caractéristiques graphiques les plus attrayantes du Commodore 64 est sa capacité d'offrir des figures. Elles sont définies, comme pour les caractères, par l'utilisateur, mais avec un format plus large de 21 lignes de 24 pixels. Les figures ne sont pas affichées sur la matrice de caractères normale de l'écran. Elles peuvent être modifiées pour un seul pixel à la fois, au lieu de huit pour déplacer un caractère d'une position à une autre. Il est possible d'afficher jusqu'à huit figures en même temps. Les caractéristiques d'une figure sont les suivantes :

Forme et couleur

Une figure est définie pratiquement comme un caractère graphique de 8 × 8 pixels. Mais 63 octets sont nécessaires pour traduire la structure codée en binaire. Une fois la forme ainsi définie, elle est stockée sur 63 positions consécutives. Chaque figure comporte un pointeur de données désignant la zone dont elle dérive. Ce qui signifie que plusieurs figures peuvent se référer à une même zone ou qu'elles peuvent être identiques. Une figure peut être modifiée en dirigeant son pointeur sur une autre zone.

Chaque figure peut être colorée dans une des 16 couleurs disponibles. Une figure peut également comporter plusieurs couleurs.

Taille et mouvement

Les figures peuvent être agrandies horizontalement ou verticalement, ou dans les deux directions, jusqu'à doubler de taille. La taille maxi-

PS1, PS2 & PS3 (– 128 à 128)

PS signifie « pitch step / palier d'intensité ». Au commencement de la note associée, pitch est paramétré par SOUND. PS1 paramètre le changement d'intensité (positif ou négatif) par palier pour la première partie de pitch envelope; PS2 s'applique à la deuxième partie; PS3, à la troisième. De manière semblable à SOUND, PS est donné en quarts de ton.

NS1, NS2 & NS3 (0 à 255)

NS, qui signifie « number of steps per section / nombre de paliers par partie », détermine en conjonction avec PS le rythme auquel l'intensité change dans une partie donnée de la gamme d'intensités, ainsi que la durée de cette dernière. PS et NS, pour l'exemple ci-dessus, sont :

```
T = 1  PS1 = – 10 NS1 = 15
       PS2 = + 10 NS2 = 10
       PS3 = – 15 NS3 = 5
```

Ici, pitch est paramétré par SOUND = 160, avec pour résultat :

ENVELOPE 1, 1, – 10, 10, – 15, 15, 10, 5, 0, 0, 0, 0, 0, 0

La durée pour ENVELOPE est : (NS1 + NS2 + NS3) × T; à savoir, ici : (15 + 10 + 5) × 1 = 0,3 seconde. Normalement, la gamme d'intensités (pitch envelope) s'autorépète sur toute la durée d'une note, à moins d'être invalidée par le paramètre de durée, T.

Nous reviendrons sur les caractéristiques sonores du BBC dans la prochaine rubrique « Son et lumière ».

male est de 48 × 42 pixels. Ici encore le prix à payer est la diminution de moitié de la résolution dans le sens de l'agrandissement. Une figure peut se déplacer un pixel à la fois, et la position quittée est automatiquement effacée. Les figures peuvent également quitter ou réintégrer la zone visible de la page-écran.

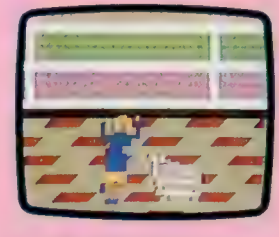
Priorité et collision

Lorsque deux figures se rencontrent, elles semblent se superposer. Si la figure de dessus comporte des trous, celle de dessous se verra à travers. Un coefficient de priorité de 1 à 7 peut être attribué aux figures afin de donner un effet visuel à trois dimensions. La priorité consiste à faire passer en premier plan les figures de niveau de priorité le plus bas. Logiquement, les figures apparaissent au-dessus des caractères normaux; mais elles peuvent être programmées pour être dessous. Cela peut également être utilisé pour donner l'impression de profondeur à l'écran.

Lorsque deux figures se croisent, l'événement est signalé au registre « Collision ». PEEK, appliqué à ce registre, peut fournir le nom des figures concernées. Un autre registre semblable signale les « collisions » avec les caractères de fond.

Ces caractéristiques rendent possible l'écriture de jeux très rapides en BASIC. Il n'existe malheureusement pas de commande spécifique BASIC pour contrôler les figures. Il faut donc faire se succéder des commandes POKE dans la mémoire du Commodore 64. Une autre méthode plus facile pour créer des figures passe par l'intermédiaire de cartouches BASIC Simon.

Cartouche Basic Simon
D'un prix abordable, cette cartouche permet l'extension de la haute résolution et des capacités graphiques. Un manuel très complet détaillant les 114 commandes est adjoint. Ces commandes comprennent la mise en exploitation du mode haute résolution, le choix des couleurs de fond et d'écriture, le tracé de cercles, d'ellipses, de rectangles et de lignes droites. Les instructions d'intervention sur les figures comprennent : l'aide à la conception, les commandes pour les faire apparaître ou pour les faire disparaître, etc.



Deuxième étape
Cette partie du programme utilise deux figures agrandies de plusieurs couleurs pour la silhouette de la ménagère, et pour le panier roulant.
Les pointeurs des données-figures sont manipulés de sorte que la ménagère change d'aspect. Ce passage d'une figure à l'autre donne l'impression que la silhouette danse.
Pour utiliser le programme « Supermarché » comme sous-programme, remplacez la ligne 3270 par 3270 RETURN.

```
100   REM **FIGURES 64**
105   PRINT " "
110   V = 53248
120   REM-----READ DONNEES FIGURE-----
130   FOR I = 12288 TO 12339 : READA : POKEI, A : NEXT
140   FOR I = 12352 TO 12414 : READA : POKEI, A : NEXT
150   FOR I = 832 TO 894 : READA : POKEI, A : NEXT
160   FOR I = 896 TO 958 : READA : POKEI, A : NEXT
170   FOR I = 12416 TO 12478 : READA : POKEI, A : NEXT
180   REM-----AGRANDIR FIGURES-----
190   POKEV + 23,7 : POKEV + 29,7
200   REM-----FIGURES COULEURS-----
210   POKEV + 39,10 : POKEV + 40,10
220   POKEV + 41,1
230   REM---PLUSIEURS COULEURS---
240   POKEV + 28,3 : POKEV + 37,7 : POKEV + 38,9
300   REM-----POINTEURS MEMOIRE-----
310   POKE 2040,192 : POKE 2041,193 : POKE 2042,194
320   REM-------DETERMINE LES COORDONNEES DE Y-------
330   YO = 150 : Y1 = YO + 42 : Y2 = YO + 84
340   POKEV + 1, YO : POKEV + 3, Y1 : POKEV + 5, Y2
400   REM-------AFFICHER FIGURES-----
410   POKEV + 21,7
500   GOSUB 3000 : REM PASSER EN L'ABSENCE DE SOUS-PROGRAMME
1000  XO = 20
1010  POKE 2040,13 : POKE 2041,14
1020  POKEV, XO : POKEV + 2, XO : POKEV + 4, XO + 48
1030  FOR I = 1 TO 500 : NEXT
1040  POKE 2040,192 : POKE 2041,193
1050  XO = XO + 5
1060  POKEV, XO : POKEV + 2, XO : POKEV + 4, XO + 48
1070  FOR I = 1 TO 500 : NEXT
1080  XO = XO + 5
1090  IF XO >  260 THEN 1110
1100  GOTO 1010
1110  FOR J = 1 TO 10
1120  POKE 2040,13 : POKE 2041,14
1130  FOR I = 1 TO 50 : NEXT '
1140  POKE 2040,192 : POKE 2041,193
1150  FOR I = 1 TO 50 : NEXT
1160  NEXT
1170  GOTO 1170
9000  REM-----DONNEES HAUT SILHOUETTE-----
9010  DATA 0, 0, 0, 0, 0, 21, 0, 0, 22, 0, 0, 85, 0
9020  DATA 0, 85, 0, 0, 85, 0, 0, 40, 0, 0, 252, 0
9030  DATA 15, 255, 0, 255, 255, 0, 255, 255, 0
9040  DATA 195, 255, 0, 195, 255, 0, 195, 243, 254
9050  DATA 207, 243, 254
9060  DATA 143, 240, 0, 143, 252, 0, 15, 252, 0
9070  DATA 15, 252, 0, 15, 252, 0
9100  REM-----DONNEES BAS SILHOUETTE-----
9110  DATA 15, 252, 0, 15, 252, 0, 15, 252, 0
9120  DATA 15, 252, 0, 5, 84, 0, 5, 84, 0, 5, 84, 0
9130  DATA 5, 84, 0, 10, 40, 0, 234, 40, 0, 234, 40, 0
9140  DATA 234, 40, 0, 192, 40, 0, 192, 40, 0, 0, 40, 0
9150  DATA 0,40, 0, 0, 63, 0, 0, 63, 0, 0, 0, 0, 0, 0, 0
9160  DATA 0, 0
9200  REM-----DONNEES HAUT SILHOUETTE #2-----
9210  DATA 0,0,0,0,20,32,32,85,32,32,105,40,40,105,40
9220  DATA 40, 105, 40, 40, 105, 40, 40, 40, 40, 40, 40, 252, 40
9230  DATA 63, 255, 240, 63, 255, 240, 63, 255, 0
9240  DATA 3, 255, 0, 3, 255, 0, 3, 240, 0
9250  DATA 15, 240, 0
9260  DATA 15, 240, 0, 15, 252, 0, 15, 252, 0
9270  DATA 15, 252, 0, 15, 252, 0
9300  REM-----DONNEES BAS SILHOUETTE #2-----
9310  DATA 15, 252, 0, 15, 252, 0, 15, 252, 0
9320  DATA 15, 252, 0, 5, 84, 0, 5, 84, 0, 5, 84, 0
9330  DATA 5, 84, 0, 10, 40, 0, 58, 168, 0, 58, 168, 0
9340  DATA 58, 0, 0, 58, 0, 0, 10, 0, 0, 10, 0, 0
9350  DATA 10, 0, 0, 15, 192, 0, 15, 192, 0, 0, 0, 0, 0, 0, 0
9360  DATA 0, 0, 0
9400  REM-----DONNEES PANIER-----
9410  DATA 192, 0, 0, 224, 0, 0, 119, 0
9420  DATA 0, 55, 192, 0, 32, 60, 0, 53
9430  DATA 87, 240, 32, 0, 15, 53, 85, 85
9440  DATA 32, 0, 3, 53, 85, 0, 0, 3
9450  DATA 21, 85, 85, 31, 255, 255, 24, 0
9460  DATA 0, 12, 0, 0, 12, 0, 0, 31, 255
9470  DATA 248, 31, 255, 255, 1, 0, 2, 7
9480  DATA 0, 14, 7, 0, 14
```

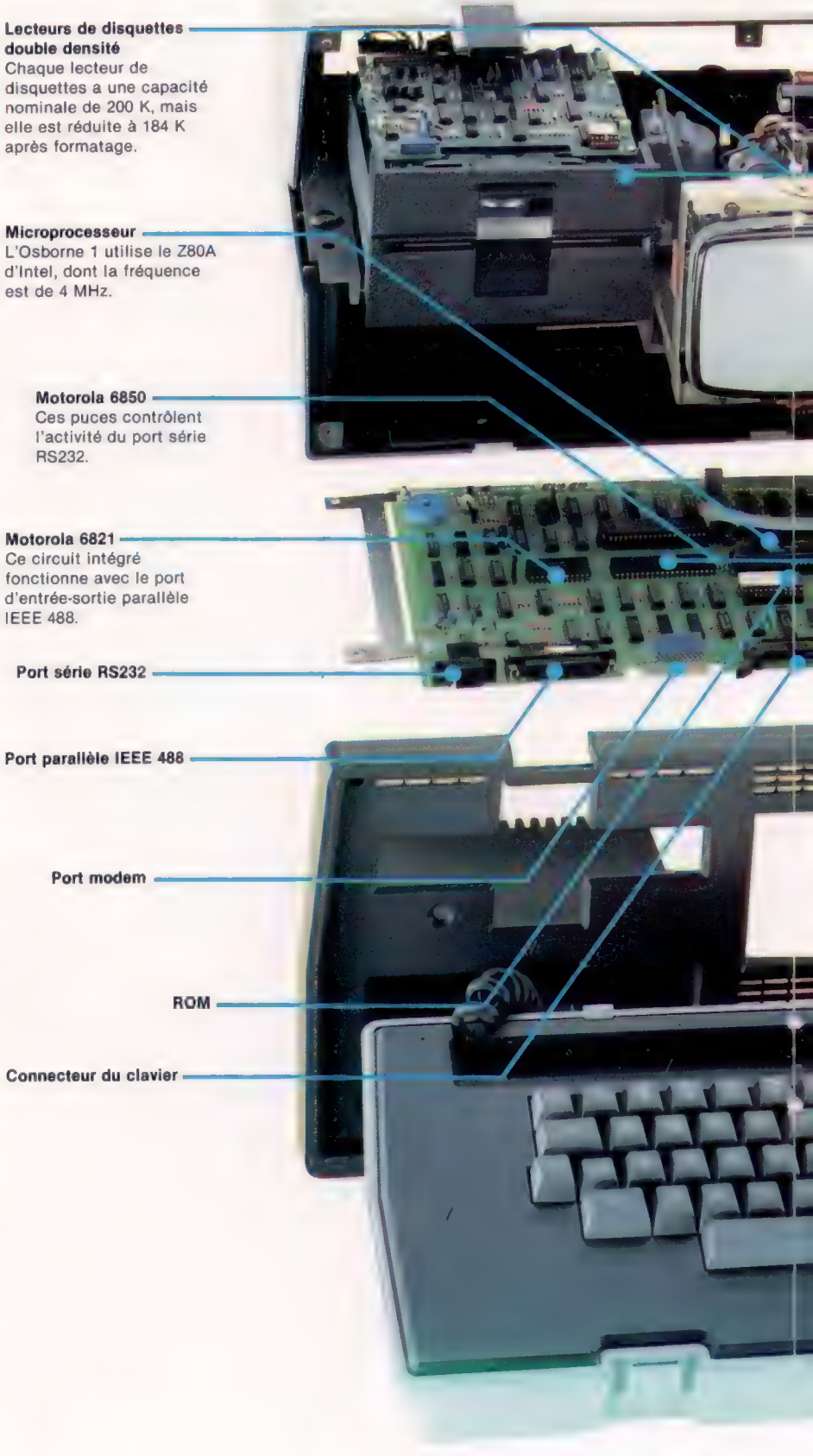
Osborne 1

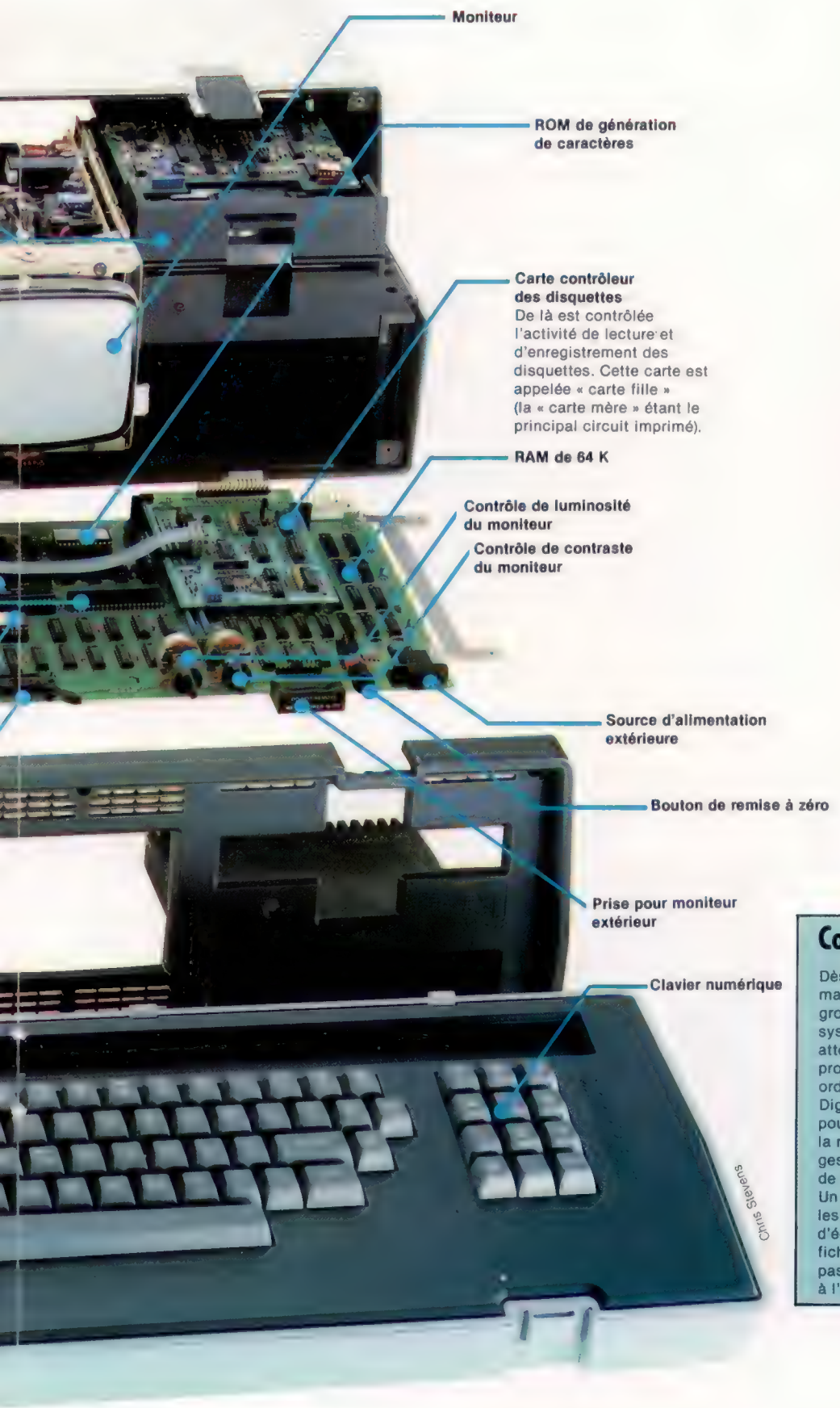
Il est le premier micro-ordinateur réellement portable, et le premier à inclure des logiciels dans son prix de vente.

L'Osborne 1 n'est pas véritablement un ordinateur domestique; mais, avec ses deux lecteurs de disquettes et son moniteur intégrés, il est et restera le premier micro-ordinateur portable entièrement autonome. Pour la première fois, en effet, à sa sortie, il était possible d'emporter avec soi son propre système de traitement de données. Il ne manquait à l'ensemble que des batteries intérieures, mais le constructeur avait pensé que cela accroîtrait le poids total (qui est déjà de 10,5 kg) au-delà des limites raisonnables. Une prise est donc installée à l'avant, avec les autres prises d'interface, et alimente l'appareil en courant continu : 12 V pour les lecteurs de disquettes, 5 V pour les circuits.

Le prix de l'Osborne 1 (en France, plus de 21 000 F) rend difficile son usage comme ordinateur familial; mais il faut noter que sont inclus dans ce prix quelques-uns des meilleurs progiciels de gestion disponibles, qui valent plusieurs milliers de francs. Ce sont : le CBASIC de Microsoft, version compilée du BASIC, qui permet une exécution bien plus rapide des programmes; Supercalc, le meilleur sans doute de la première génération des tableurs (tableaux de calculs financiers); Wordstar et Mailmerge, deux traitements de texte transférables (non limités à un seul type de machine) parmi les plus cotés; et surtout le système d'exploitation CP/M (Control Program/Monitor), dû à Digital Research, qui, une fois adopté, permet l'usage d'un grand nombre de logiciels.

Sur l'Osborne 1, comme sur l'Apple II, le système d'exploitation est chargé à partir d'une disquette. Le CP/M supervise le fonctionnement général de l'ordinateur et permet l'accomplissement direct d'opérations de service internes : initialisation de nouvelles disquettes, repérage de leur contenu, copie de fichiers... Il possède encore d'autres points forts. Les logiciels sont écrits pour lui, et non pour les machines qui l'emploient. Le marché potentiel est donc plus grand; les firmes qui produisent des programmes peuvent ainsi investir davantage d'argent dans leur mise au point et leur production, ce qui ne peut qu'accroître la qualité du produit final. Par ailleurs, le type de machine utilisé est sans importance pour un utilisateur informé; ce matériel peut donc être amélioré ou remplacé, ce qui rend inutiles la conversion des programmes et la réinsertion des fichiers de données, deux opérations toujours onéreuses. Pendant une brève période, Osborne offrit même en supplément le dBase II d'Ashton-Tate, le plus puissant de tous les programmes de





OSBORNE 1

PRIX
21 000 F environ.

DIMENSIONS
510 × 325 × 225 mm.

POIDS
10,5 kg.

UC
Z80A.

FRÉQUENCE DE L'HORLOGE
4 MHz.

MÉMOIRE
64 K de RAM.
4 K de ROM.

AFFICHAGE VIDÉO
24 rangées de 52 caractères.

INTERFACES
RS232, IEEE, Modem.

LANGAGES FOURNIS
BASIC, Assembleur du Z80.

AUTRES LANGAGES DISPONIBLES
Ce qui tourne sous CP/M.

LIVRÉ AVEC
CP/M, Wordstar, CBASIC, MBASIC, Mailmerge, Supercalc, manuels.

CLAVIER
69 touches de type machine à écrire, plus un clavier numérique.

DOCUMENTATION
Adam Osborne vendit sa maison d'édition à McGraw-Hill pour financer la construction de ses ordinateurs ; il n'est donc pas surprenant que le manuel d'utilisation soit de très haute qualité.

Control Program/Monitor

Dès l'apparition de la deuxième génération de machines, vers le milieu des années soixante, les gros et les moyens ordinateurs purent disposer de systèmes d'exploitation indépendants, mais il fallut attendre une douzaine d'années avant que ces programmes de contrôle soient accessibles sur micro-ordinateurs. Le CP/M (Control Program/Monitor) de Digital Research fut le premier d'entre eux. Conçu pour le 8080 d'Intel et le Z80 de Zilog, il autorise la mise en œuvre de programmes utilitaires, ou de gestion interne, et permet aussi d'interrompre et de faire repartir un programme mis en route.
Un autre avantage fondamental est qu'il peut redéfinir les structures. Par l'intermédiaire d'un programme d'échanges comme BSTAM, qui réduit n'importe quel fichier à sa plus simple expression, on peut donc faire passer un programme écrit pour CP/M d'une machine à l'autre, quelles que soient leurs caractéristiques.

bases de données financières, qui se vend d'ordinaire pour plusieurs milliers de francs.

Malheureusement, tous les milieux d'affaires américains n'avaient d'yeux que pour l'IBM PC, un ordinateur 16 bits construit autour du microprocesseur 8088 d'Intel. Simple solution d'attente (qui se flatte d'avoir 16 bits à l'adressage, mais n'a que 8 bits pour la transmission des données), le 8088 devint un standard de fait pour la simple raison qu'IBM l'avait choisi pour faire son entrée sur le marché des micro-ordinateurs.

L'IBM PC possède un système d'exploitation qui lui est propre, le PC-DOS. Voulant relever le défi, Digital Research mit en vente deux nouvelles versions du CP/M : le CP/M Concurrent, qui permet une communication réelle entre utilisateurs multiples; et le CP/M86, conçu pour le microprocesseur 8086 d'Intel : 16 bits à l'adressage, 16 bits pour la transmission des données.

Il était pourtant trop tard pour Osborne, écrasé par les impitoyables lois du marché, et en 1983 l'Osborne Computer Corporation (la compagnie mère américaine) se mit en liquidation volontaire. Aujourd'hui encore, avec ses 64 K de RAM (dont 60 accessibles à l'utilisateur) et ses deux lecteurs de disquettes d'une capacité de 183 K chacun, l'Osborne 1 reste une machine assez puissante. Avec ses différents ports (RS232, IEEE, modem) et son alimentation par simple batterie, il n'est pas très difficile de comprendre pourquoi elle eut tant de succès et reste si appréciée de ses possesseurs, même après la faillite de son créateur.

Une caractéristique très intéressante de l'Osborne 1, qu'il partage en partie avec l'HX-20 d'Epson, est l'existence d'un « écran virtuel », qui est plus de trois fois plus large que l'affichage standard (24 rangées de 52 caractères). L'emploi d'une touche CONTROL (obligatoire sous CP/M) et des curseurs permet d'afficher l'ensemble de la mémoire d'écran. Cela permet de surmonter presque tous les inconvénients liés à la petite taille de l'écran (8,75 × 6,6 cm), même si les profanes sont souvent surpris que des caractères de 2 mm de haut puissent être agréables d'emploi et tout simplement lisibles!

En fait, cette miniaturisation est rarement un obstacle insurmontable, bien qu'Osborne ait pris soin d'installer une prise qui permet le raccord de l'appareil à un moniteur extérieur plus grand. Loin d'être gênés, beaucoup d'utilisateurs réclamèrent un affichage permanent des 4 K de la mémoire d'écran (32 rangées de 128 caractères), et la machine fut modifiée dans ce seul but. On peut ainsi sélectionner trois « largeurs » d'écran, de 52, 96 et 128 caractères, et ceux-ci, même dans le dernier cas, sont toujours lisibles et bien dessinés.

Le clavier de l'Osborne 1, qui se fixe à l'avant de l'appareil (il lui sert de couvercle et le met à l'abri des intempéries), comporte 69 touches de type machine à écrire. Viennent s'y ajouter les touches CONTROL et ESCAPE, plus un clavier numérique situé sur la droite, qui comporte les touches « point » et ENTER. A l'aide d'un programme CP/M intitulé Setup, il est possible de redéfinir l'usage des touches numériques (en conjonction avec la touche CONTROL), jusqu'à concurrence de 96 caractères. Cela est très utile lorsqu'un mot, une phrase ou une instruction reviennent fréquemment. Les modifications ainsi obtenues ne sont pas stockées en mémoire, mais sur disquette; elles peuvent donc être adaptées à chaque logiciel particulier, et mises en route chaque fois que le système d'exploitation est activé.

En plus des 96 caractères standard (majuscules et minuscules), il existe 32 caractères graphiques prédéfinis, mais on ne peut les mettre en œuvre que par l'intermédiaire d'un programme d'application.

L'Osborne ayant recours à des puces de la famille du 6800 (et non du 8080, comme on aurait pu s'y attendre de la part d'une machine qui fonctionne sous CP/M), l'interrogation du clavier se fait de façon un peu différente. La ROM du microprocesseur vérifie sans cesse si une touche a été enfoncée; si oui, une part de la mémoire établit laquelle. Il n'y a aucun système de décodage dans le clavier lui-même. Cela permet une programmation aisée des touches de fonction; les instructions étant stockées dans la RAM, il est facile d'y accéder et de les modifier de l'intérieur même d'un programme.

Il n'en reste pas moins que bon nombre de professionnels — de l'agent commercial à l'ingénieur — qui sont appelés à se déplacer très souvent ou doivent rester pratiquement toujours en liaison avec leur entreprise gardent une grande confiance dans les qualités de l'Osborne 1 (et de son grand frère l'Osborne Executive). Son intérêt reste indéniable. La meilleure preuve est donnée par la filiale britannique d'Osborne qui continue ses activités après avoir repris son autonomie.

Le transport
En plus de son excellent rapport qualité/prix, l'Osborne 1 a l'avantage d'être portable. Ses 10,5 kg ne sont pas négligeables, mais, autonome et bien équilibré, il peut être transporté sans grands problèmes. Dès le départ il était prévu pour pouvoir être glissé sous un fauteuil d'avion.

(Cl. Ian McKinnell.)

Tri de code

Le tri Shell est plus efficace que le tri par paires ou que le tri par insertion. Il concerne de grands tableaux et procède en divisant les données en une suite de chaînes.

Nous avons déjà fait état de deux méthodes de tri : le tri par paires qui est plus facile, et le tri par insertion, plus rapide. L'expérience de ces deux méthodes montre que ce qui prend du temps est d'échanger les cartes sur de courtes distances. Il est préférable d'attendre de pouvoir trier une seule fois sur une longue distance plutôt que plusieurs fois sur de courtes distances.

```
7799 REM*****************************************
8000 REM*                                  SHELL
8001 REM*****************************************
8025 PRINT « TRI DE TYPE SHELL - GO!!! »
8050 LET LK = LT
8100 FOR Z = O TO I STEP O
8150 LET LK = INT (LK/II)
8200 FOR LB = I TO LK
8250 LET LL = LB + LK
8300 FOR P = LL TO LT STEP LK
8350 LET D = R(P)
8400 FOR Q = P TO LL STEP - LK
8450 LET R(O) = R(Q - LK)
8500 IF D < = R(O) THEN LET R(Q) = D : LET Q = LL
8550 NEXT Q
8600 IF D > R(LB) THEN LET R(LB) = D
8650 NEXT P
8700 NEXT LB
8750 IF LK = I THEN LET Z = I
8800 NEXT Z
8850 PRINT « TRI DE TYPE SHELL - STOP!!! »
8900 RETURN
```

Pour ajouter cette routine au programme de tri type vu précédemment, modifiez ainsi la ligne 350

```
350 LET I = 1 : LET O = 0 : LET II = + : LET TH = 3
```

et changez ainsi la ligne 900

```
900 ON SR GOSUB 6000, 7000, 8000
```

Le tri Shell (du nom de son créateur) est meilleur que les deux précédents. Cette méthode vise à réduire le désordre du tableau très tôt dans le tri. Les éléments se retrouvent près de leur position voulue. Les échanges peuvent se faire sur des distances relativement longues. Voici une méthode d'application de ce type de tri :

1° Abattez les cartes à trier sans vous préoccuper de leur ordre. Elles doivent être triées par ordre décroissant de sorte que le roi soit la carte la plus à gauche, et l'as la carte la plus à droite. Comptez les cartes, divisez ce nombre (13 ici) par deux, et, sans vous préoccuper du reste, inscrivez le résultat (6 ici) sur un morceau de papier appelé le « lien ».

2° Placez une pièce de 10 centimes sous la carte la plus à gauche (dénommée position 1), et une pièce de 50 centimes sur le lien (en sixième position ici). Toutes les cartes comprises entre la première carte et le lien doivent être individuellement la plus à gauche d'une suite de chaînes de cartes. Le nombre de chaînes sera celui du lien. Chaque chaîne commence par sa dernière carte, et progresse en ajoutant le lien au numéro de la position de la dernière carte, pour obtenir la carte suivante, et ainsi de suite jusqu'à ce que le tableau soit épuisé. La première chaîne comporte donc les cartes dans les positions Un, Sept et Treize. La deuxième contient les positions Deux et Huit; la troisième Trois et Neuf. La dernière, Six (la valeur du lien pour cet exemple) et Douze.

3° Après avoir indiqué les limites des cartes avec les pièces de monnaie, retirez les cartes qui forment la première chaîne, et triez-les avec l'une des méthodes décrites plus haut (le listage est ici trié par insertion).

4° Réinsérez la chaîne triée dans les vides, et recommencez l'opération précédente avec la chaîne suivante, et encore la suivante jusqu'à ce que toutes les chaînes dont la carte la plus à gauche se trouve entre les deux pièces aient été triées.

5° Lorsque toutes les chaînes sont triées, divisez le lien par deux, en ignorant le reste. Si le lien est inférieur à 1, la chaîne sera considérée comme triée. Dans le cas contraire, répétez l'opération décrite plus haut avec la nouvelle valeur du lien.

Dans l'exemple ci-dessous, le tri Shell s'est révélé le plus efficace au-delà de 40 articles.

Panneau de tri de type Shell

N° position	Valeur lien	Commentaires
1 2 3 4 5 6 7 8 9		
2 8 9 3 T 5 K 6 7	(9/2) = >4	Commencer passage
* +ⓐ$ * +ⓐ$ *		Création des chaînes
T 7 2		Tri chaîne 1
8 5		Tri chaîne 2
K 9		Tri chaîne 3
6 3		Tri chaîne 4
T 8 K 6 7 5 9 3 2		Commencer passage
T 8 K 6 7 5 9 3 2	(4/2) = >2	Fin passage
* + * + * + * + *		Création chaînes
K T 9 7 2		Tri chaîne 1
8 6 5 3		Tri chaîne 2
K 8 T 6 9 5 7 3 2		Fin passage
K 8 T 6 9 5 7 3 2	(2/2) = >1	Commencer passage
* * * * * * * * *		Création chaînes
K T 9 8 7 6 5 3 2		Fin de passage

Clef des signes	
*	Élément de chaîne 1
+	Élément de chaîne 2
ⓐ	Élément de chaîne 3
$	Élément de chaîne 4

Tri Shell
Cet exemple portant sur un nombre réduit de cartes montre comment le tableau est divisé en chaînes (avec des vides fondés sur la valeur courante du lien). Les chaînes sont triées séparément, dans ce cas qui utilise la méthode d'insertion, avant la fin d'un passage.
Le listage correspondant du tri Shell donné ici doit être ajouté au programme banc d'essai de la page 287.

D'une seule main

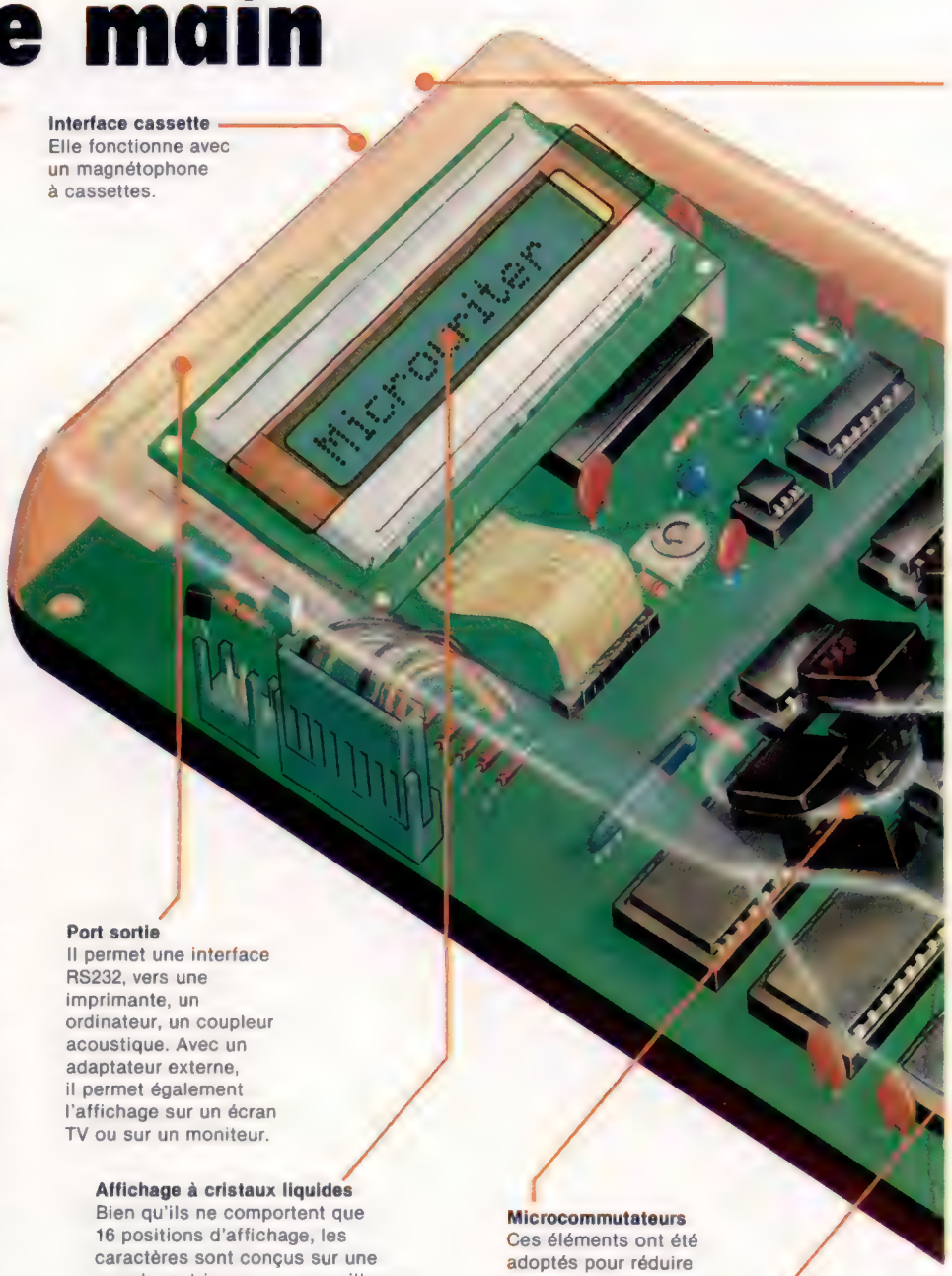
Le Microwriter est un système de traitement de texte portable manœuvré d'une seule main. Le clavier à six touches est révolutionnaire.

Un traitement de texte professionnel ou un ordinateur familial muni d'un programme de traitement de texte est très utile. En dehors du fait qu'ils libèrent des besognes fastidieuses d'écriture et de paperasserie, les traitements de texte servent à la documentation de programmes, reproduisent immédiatement les circulaires, ou manipulent efficacement un répertoire. Ils peuvent devenir indispensables à toute forme d'écriture en remplaçant le papier et le stylo. Mais un problème se pose quand il s'agit de prendre des notes chez vous ou au travail de telle manière que l'ordinateur puisse les comprendre par la suite.

Les ordinateurs portables représentent un marché en pleine expansion. Des systèmes tels que le Tandy Model 100, ou l'Epson HX-20 peuvent être utilisés comme traitements de texte portables, ou comme terminaux de systèmes éloignés et plus importants. Ils sont aussi maniables qu'un bloc-notes ou un dictaphone. Que dire alors d'un traitement de texte de poche? Un système tellement performant qu'il est autonome et s'utilise d'une main, tout en étant susceptible d'être relié à une imprimante ou à un micro-ordinateur!

Il s'agit du Microwriter, disponible depuis quatre ans sur le marché. Mis au point par Cy Endfield, Américain d'origine, cet appareil abandonne le clavier traditionnel au profit d'un système ne comportant que six touches. Le concept a d'abord été conçu en vue d'un jeu tenant dans la main et portant sur des mots. Le clavier habituel, même miniaturisé, aurait été trop gros et trop coûteux. La solution a été de construire un clavier dont le nombre très réduit de touches regroupe tous les caractères alphanumériques. Un système de codage symbolique propre au Microwriter a résolu la question.

A première vue, il semble impossible que le Microwriter puisse générer du texte. En fait, les combinaisons de ces six touches reproduisent tous les caractères, y compris les caractères de ponctuation et les chiffres. La pratique de ces combinaisons s'acquiert en quelques heures seulement. Le clavier est beaucoup plus maniable qu'un clavier normal. La combinaison des touches destinées à représenter une lettre est fondée sur la ressemblance physique avec la lettre, sur sa forme. La frappe s'adresse ainsi à des non-professionnels. Le fait qu'une seule main soit nécessaire rend également ce système particulièrement indiqué aux personnes handicapées qui ne peuvent manipuler toutes les touches d'un

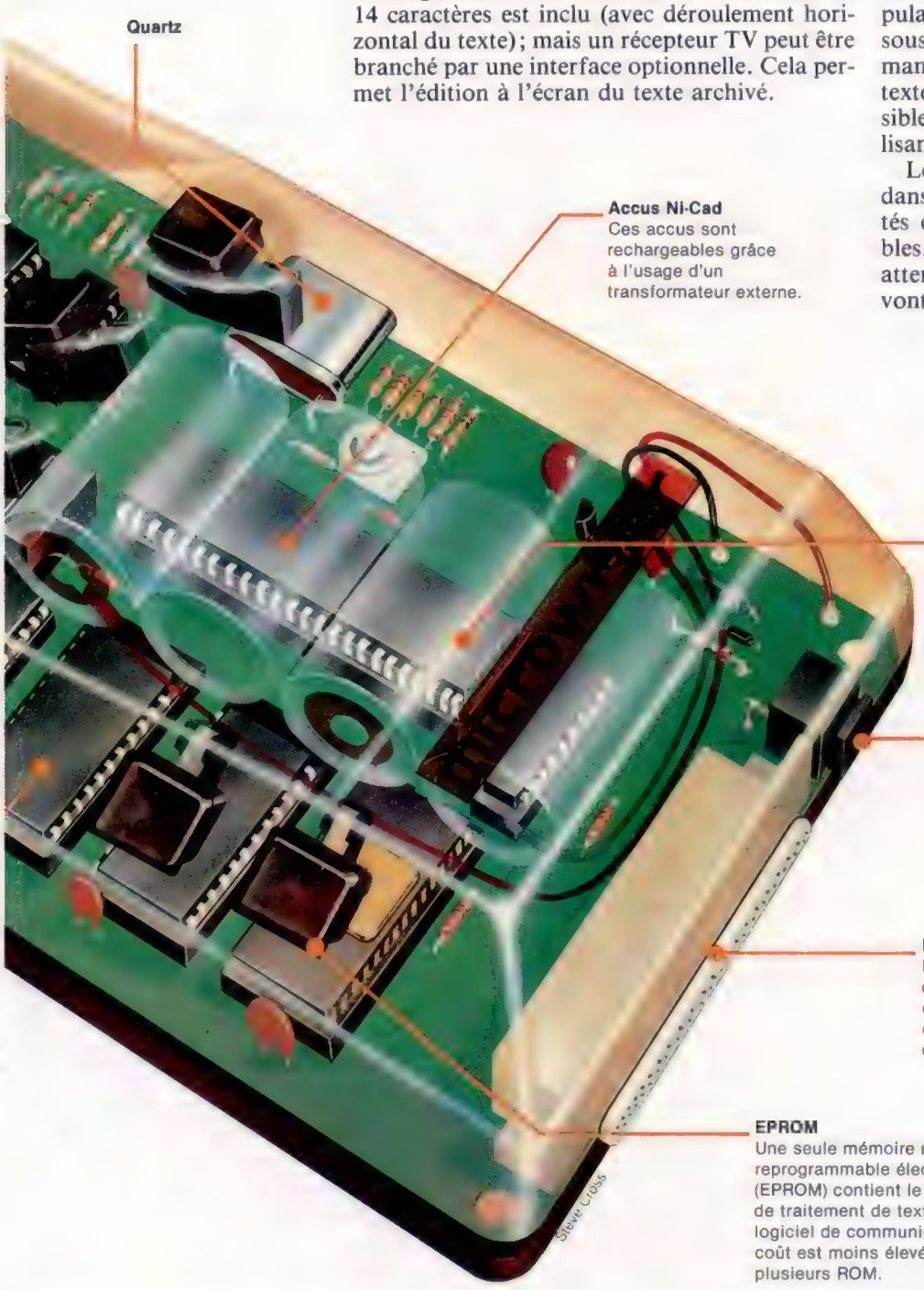


clavier classique pour introduire des instructions.

Du point de vue interne, le Microwriter est conçu pour être tout à fait portable. Le microprocesseur interne et la mémoire sont à technologie CMOS, réduisant ainsi la consommation d'énergie. L'autonomie de la pile au Ni-Cad rechargeable est de trente heures. Un écran de 14 caractères est inclu (avec déroulement horizontal du texte); mais un récepteur TV peut être branché par une interface optionnelle. Cela permet l'édition à l'écran du texte archivé.

L'interface série RS232 est destinée à une imprimante, mais il existe aussi une interface cassette pour le stockage. L'interface série transforme également le Microwriter en terminal de micro-ordinateur d'un système de traitement de texte. La transmission des documents est ainsi assurée en vue de leur traitement plus complet, c'est-à-dire une édition ou une manipulation. Le Microwriter peut stocker du texte sous forme de documents distincts. Des commandes d'édition sommaires sont prévues. Du texte peut être ajouté ou supprimé, et il est possible de déplacer de grands blocs de texte en utilisant l'interface cassette comme tampon.

Le Microwriter a été conçu pour être intégré dans des systèmes plus larges avec d'autres unités électroniques. Malgré ses qualités indéniables, son succès reste encore limité. Il faut donc attendre pour voir si des constructeurs de micro vont reprendre son principe.



Interrupteur on/off
L'arrêt de la machine n'entraînera pas la perte du document, et l'on peut le réintégrer instantanément.

Quartz

Accus Ni-Cad
Ces accus sont rechargeables grâce à l'usage d'un transformateur externe.

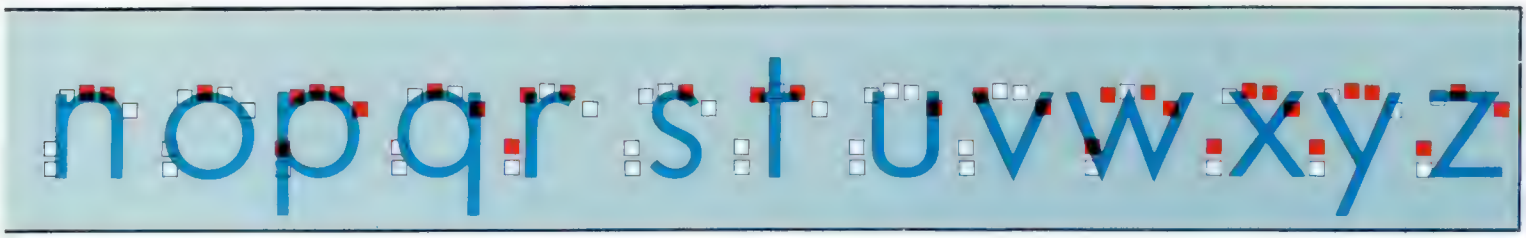
Unité centrale
Mémoire centrale et mémoire vive sont en technologie CMOS pour réduire la consommation d'énergie.

Connecteur d'alimentation
Pour recharger les accus ou pour brancher sur le secteur par un transformateur externe.

Interface d'extension
Destiné aux futures extensions, ce port comporte l'adresse du microprocesseur et des lignes de données.

EPROM
Une seule mémoire morte reprogrammable électriquement (EPROM) contient le programme de traitement de texte et le logiciel de communications. Son coût est moins élevé que plusieurs ROM.

Documentation
La documentation du Microwriter comprend des mnémoniques et des illustrations afin d'aider l'utilisateur à acquérir l'usage des combinaisons de touches nécessaires à la création de l'alphabet. La sixième touche est utilisée conjointement aux cinq autres pour les caractères de ponctuation et les commandes d'édition.



(Liz Dixon)

Mandat de perquisition

Le temps pris pour localiser un enregistrement précis peut être très réduit en utilisant la « recherche dichotomique » — pourvu que le fichier ait déjà été trié dans un ordre adéquat.

Les trois activités essentielles dans le programme de carnet d'adresses — ajouter de nouveaux enregistrements, sauvegarder le fichier sur bande ou disque et lire le fichier dans la mémoire de masse la première fois que le programme tourne — ont été expliquées. Mais un carnet d'adresses ne sert à rien si l'on ne peut qu'y ajouter de l'information et pas en extraire. En outre, il faut une routine pour trouver un enregistrement.

L'activité la plus fréquente est probablement de trouver un enregistrement complet à partir d'un nom. C'est pourquoi la première option du menu de choix (*CHOIX*) est TROUVER ENREGISTREMENT (A PARTIR DU NOM). La recherche est une activité extrêmement importante dans beaucoup de programmes informatiques, surtout dans les programmes de bases de données où l'on a souvent besoin de retrouver des articles spécifiques. En gros, il y a deux méthodes de recherche : linéaire et dichotomique. Une recherche linéaire considère chaque élément d'un tableau, en commençant par le début, jusqu'à ce que l'article particulier soit localisé. Si les données ne sont pas triées dans le tableau, la recherche linéaire est la seule qui marchera à coup sûr. Le temps pour localiser l'article en utilisant cette méthode dans un tableau de N articles aura une valeur moyenne proportionnelle à N/2. Si les articles sont peu nombreux, N/2 peut parfaitement être acceptable, mais, souvent, le temps nécessaire pour la recherche peut devenir excessif.

Si les données du fichier sont triées, il existe une méthode de recherche bien plus efficace, connue sous le nom de « recherche dichotomique ». Supposons que vous vouliez trouver la définition du mot « leptocéphale » dans un dictionnaire. Vous n'allez pas commencer par la première page pour voir s'il y est, puis passer à la deuxième, et ainsi de suite jusqu'à ce que vous le trouviez. Au lieu de cela, vous mettez votre pouce à peu près au milieu du livre, ouvrez la page et regardez ce qui est écrit. Si la page commence par « métatarsien », vous savez que vous êtes allé trop loin, donc la seconde moitié du livre ne convient pas et le mot que vous voulez se trouvera quelque part dans la première. Puis vous répétez le processus, traitant la page ouverte comme s'il s'agissait de la fin du dictionnaire. Vous partagez à nouveau la première partie du dictionnaire en deux et ouvrez pour trouver « dolic ». Cette fois, vous savez que la page choisie est trop « bas » et (pour les besoins de notre recherche de « leptocéphale ») peut être considérée comme la première page — toutes les pages précédentes étant trop « bas » dans le sens ou « l » est plus « haut » que « d ». La « première » et la « dernière » page du dictionnaire peuvent maintenant être considérées comme celles commençant respectivement par « dolic » et « métatarsien ». Vous remettez votre pouce au milieu de la partie « adéquate » et ouvrez à « kératine ». C'est encore trop « bas », donc le mot que nous cherchons doit se trouver entre cette page et celle de « métatarsien ». En répétant ce processus suffisamment souvent, on est sûr de localiser le mot cherché — à condition qu'il soit dans le dictionnaire !

La routine de recherche nécessaire pour notre carnet d'adresses devra être beaucoup plus compliquée qu'elle n'apparaît de prime abord, pour des raisons qui deviendront évidentes. La première chose que fera la routine de recherche — appelons-la *RECHENR* pour le moment — est de demander le nom à chercher. C'est la clé de recherche. Supposons qu'il y ait quelque part dans le fichier un enregistrement pour une personne appelée Pierre Durand. L'enregistrement en question comprend une zone contenant DURAND PIERRE. La routine de recherche nous pose une question telle que QUI CHERCHEZ-VOUS ?, et nous répondons PIERRE DURAND, ou, peut-être, P. DURAND. Pour simplifier, supposons que nous répondions par le nom complet, PIERRE DURAND. La première mission de la routine de recherche sera de convertir cette réponse dans sa forme standardisée, DURAND PIERRE. Ensuite, elle comparera notre donnée, la clé de recherche, avec les divers contenus de la zone MODNOM$. Si le programme utilisait une recherche linéaire, la clé de recherche serait comparée à chaque zone MODNOM$ successivement jusqu'à ce qu'on trouve l'existence ou la non-existence d'une correspondance.

Comme nous l'avons déjà noté, une recherche linéaire n'est pas efficace en comparaison d'une recherche dichotomique si les données sont déjà triées. La routine de recherche peut vérifier que les enregistrements sont triés en commençant par IF RMOD = 1 THEN GOSUB *TRIENR*. Le programme sait que l'élément inférieur du tableau à chercher est MODCHP$(1) et le supérieur MODCHP$(TAILLE – 1). Pour mener la recherche, il nous faudra trois variables : INF pour le bas du tableau (MODCHP$(1) au commencement); SUP pour le haut du tableau (MODCHP$(TAILLE – 1) au commencement); et MED pour la valeur correspondant à l'élément médian.

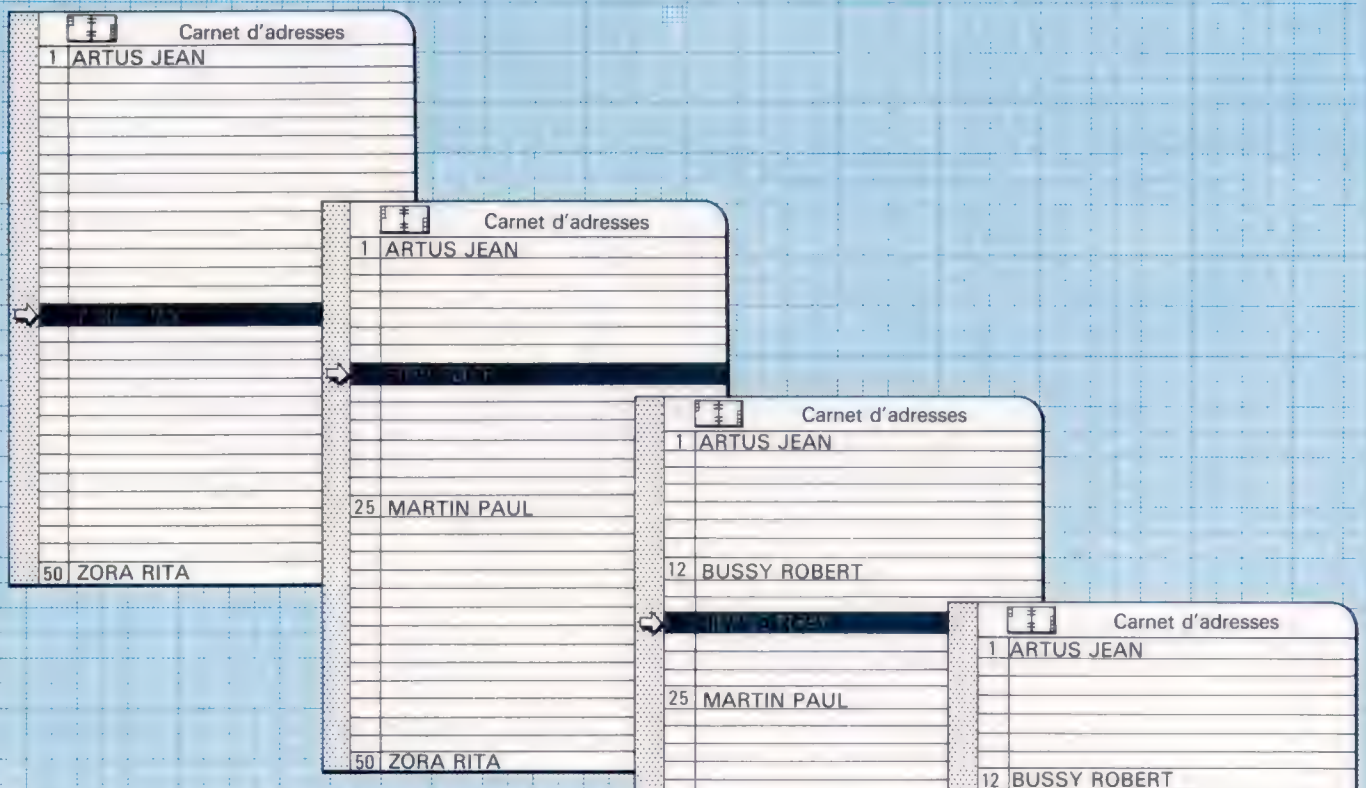
Par analogie avec le dictionnaire, nous pouvons supposer que INF = TABLEAU(1) et SUP = TABLEAU(TAILLE – 1). Autrement dit, le tableau que nous devons considérer pour la recherche commence par le plus « petit » élément et se termine par le plus « grand ». C'est pourquoi nous pouvons faire LET INF = 1 et LET SUP = TAILLE – 1 (rappelez-vous que (TAILLE) est toujours supérieur d'une unité au nombre d'enregistrements).

Admettons qu'il y ait 21 entrées valables dans le carnet d'adresses. TAILLE sera égal à 22, INF à 1 et SUP à 21. La valeur de MED, position de l'élément médian, peut être calculée en BASIC par INT ((INF + SUP)/2). Si la valeur de INF est 1 et celle de SUP 21, MED sera égal à 11.

Pour mener une recherche dichotomique, nous supposons d'abord que tout le fichier est valable et trouvons le milieu INT((INF + SUP)/2) à l'intérieur d'une boucle qui se termine si la cible est atteinte ou s'il n'existe pas de correspondance. Puis nous cherchons si la clé de recherche (CLERECH$) est égale à la valeur MED du tableau. Si MED est trop petit, nous savons que TABLEAU(MED) est la partie inférieure du tableau qu'il nous faut considérer, et donc MED devient INF. Toutefois, il est un peu plus efficace de poser INF égal à MED + 1, puisque nous savons déjà que TABLEAU(MED) n'est pas égal à la clé de recherche. De même, si TABLEAU(MED) > CLERECH$, on pose SUP égal à MED – 1.

Avant de développer l'ensemble de la routine, le programme peut accepter une donnée factice (qui doit avoir exactement le même format que la zone MODCHP$) et imprimer soit ENREGISTREMENT NON TROUVÉ s'il n'y a pas de correspondance, soit L'ENREGISTREMENT EST LE NO (MED) s'il y a correspondance. Comme la routine commence à la ligne 13000, elle peut être ajoutée à la fin du pro-

Recherche d'adresse



Recherche dichotomique

Si un fichier d'adresses contient 50 entrées, triées alphabétiquement de ARTUS à ZORA, alors la position pour commencer à chercher une entrée particulière (ici DURAND PIERRE) sera à mi-chemin entre les deux extrêmes. Voilà le principe sur lequel est fondée la recherche dichotomique.

La position 25 contient MARTIN PAUL, qui vient après DURAND PIERRE, ce qui indique que la seconde moitié du fichier peut désormais être ignorée dans la recherche.

En examinant le reste de la liste, nous obtenons la position 12, qui contient BUSSY ROBERT ce qui indique que DURAND PIERRE doit se trouver entre les éléments 13 et 24. Donc, la prochaine position de recherche sera 18 (DIVA ALICE), suivie de 21. Et là, nous avons atteint notre cible, DURAND PIERRE. (Cl. Kevin Jones.)

gramme et elle fonctionnera dans la mesure où l'on change la ligne 4040 en IF CHOI = 1 THEN GOSUB 13000.

La ligne 13240 contient l'instruction STOP. Elle arrête le programme temporairement dès que les messages ENREGISTREMENT NON TROUVÉ ou L'ENREGISTREMENT EST LE NO (MED) sont affichés. Le programme peut repartir au même numéro de ligne, sans perdre de données, si l'on tape CONT. Sans STOP, le programme se précipiterait sur l'instruction RETURN à la ligne 13250 et le message apparaîtrait trop brièvement pour être lisible.

La ligne 13100 donne à INF la valeur 1, position de l'élément le plus bas dans le tableau MODCHP$. SUP prend la valeur TAILLE – 1 à la ligne 13110. C'est la position des tableaux MODCHP$ où l'élément supérieur est localisé. La ligne 13120 initialise une boucle qui ne se terminera que lorsqu'une correspondance sera trouvée ou bien lorsque l'on saura qu'il n'en existe pas.

La ligne 13130 trouve le point médian du tableau en divisant par deux la somme des indices inférieur et supérieur du tableau (INT sert à arrondir la division, afin que MED ne puisse pas prendre une valeur telle que 1,5). Il est possible que le contenu de MODCHP$(MED) soit le même que la clé de recherche (CLERECH$), mais comme ce n'est pas le cas, on fera L égal à 0, en s'assurant que la boucle sera répétée. Si le test de la ligne 13140 n'est pas vérifié, MODCHP$(MED) sera soit inférieur, soit supérieur à CLERECH$. On posera alors INF égal à l'ancienne valeur de MED plus un (ligne 13150), ou bien SUP égal à l'ancienne valeur de MED moins un. La raison pour laquelle la valeur de MED elle-même ne sert pas est que le test de la ligne 13140 a déjà démontré que MODCHP$(MED) n'était pas la cible que nous cherchions et qu'il n'y a pas lieu de considérer cet élément du tableau au prochain passage de la boucle.

Si l'on ne trouve aucune correspondance, la valeur de INF pourra finalement être supérieure à celle de SUP. La boucle peut s'achever (ligne 13170) et le message ENREGISTREMENT NON TROUVÉ s'affiche (ligne 13200).

Ce fragment de programme est présenté en guise de démonstration et pour permettre de tester la routine de recherche. Tel quel, son usage est assez limité. Sans le STOP à la ligne 13240, nous n'aurions même pas le temps de voir le message sur l'écran. Ce qu'il faut, c'est l'affichage de tout l'enregistrement, tel qu'il a été entré en machine. Le numéro d'enregistrement connu, il est facile de récupérer n'importe laquelle des informations cherchées — NOMCHP$, RUECHP$, etc. A la suite de l'affichage de l'enregistrement, nous aimerions sans doute avoir un message tel que APPUYER SUR LA BARRE D'ESPACEMENT POUR CONTINUER (retour au menu principal) et peut-être d'autres options comme TAPER « P » POUR IMPRIMER.

Il n'est malheureusement pas si facile de décider comment s'y prendre pour entrer *TROUVENR*. Dans le fragment de programme, les données en entrée (à la ligne 13020) doivent être sous la forme standard — DURAND PIERRE, par exemple. Ce n'est évidemment pas l'idéal. On ne pense pas à des noms dans l'ordre inverse, ni à écrire ces noms en majuscules. De plus, la moindre différence avec le nom entré à l'origine provoquerait l'apparition du message ENREGISTREMENT NON TROUVÉ. On pourrait croire que les deux premiers problèmes seront résolus par *MODNOM*. Le troisième, consistant à trouver des correspondances approchées, est bien plus intéressant, mais beaucoup plus difficile à résoudre.

Avant de considérer ce problème, voyons pourquoi *MODNOM* ne résoudra pas les deux premiers problèmes. En se référant à *MODNOM* qui commence à la ligne 10200, on découvre une bonne illustration de l'un des pièges les plus communs dans lesquels tombent les programmeurs — le manque de généralité. Ce sous-programme devrait pouvoir convertir des noms « normaux » en noms « standard » si nécessaire. Même s'il a été écrit comme un sous-programme séparé, il était implicitement associé à *AJOUTENR*. Cela suppose que le nom à convertir réside toujours dans NOMCHP$(TAILLE) et que, après conversion, le nom modifié soit toujours stocké dans MODCHP$(TAILLE). Dans cette situation, le programmeur a trois possibilités : soit réécrire complètement *MODNOM* pour le rendre plus général, ce qui entraîne des changements dans d'autres parties du programme; soit écrire une routine presque identique pour introduire *TROUVENR*, ce qui représente un travail superflu et occupe plus de place en mémoire; soit recourir à une mauvaise technique de programmation permettant d'utiliser la routine *MODNOM* telle quelle. Cette dernière possibilité est la moins séduisante. Elle résoudra le problème, mais rendra confus le fonctionnement de la partie du programme qui a été modifiée, même pour celui qui a écrit le programme.

La morale de l'histoire est qu'il faut faire des sous-programmes aussi généraux que possible, afin qu'ils puissent être appelés par n'importe quelle partie du programme.

Pour illustrer une mauvaise technique de programmation et montrer comment cela peut rendre le programme confus, considérons la ligne 13020 du fragment de programme, INPUT « CLÉ DE RECHERCHE »; CLERECH$ et voyons la modification qui nous permettrait d'utiliser *MODNOM* :

```
13020 INPUT « CLÉ DE RECHERCHE »; NOMCHP$(TAILLE)
13030 GOSUB 10200 : REM *MODNOM*
  SOUS-PROGRAMME
13040 LET CLERECH$ = MODCHP$(TAILLE)
13050 ...
```

Heureusement TAILLE est toujours plus grand d'une unité à l'enregistrement supérieur. Autrement dit, il n'y a pas d'enregistrement à la position TAILLE dans les tableaux, de sorte que la modification ne concernera aucun enregistrement existant. Mais s'il n'y avait pas d'explication sous forme de REM, ces trois lignes paraîtraient confuses à quelqu'un d'extérieur au programme. Revenons au problème plus intéressant des

correspondances approchées. Supposons que nous ayons entré le nom PIERRE DURANT lors de la procédure *AJOUTENR*, et PIERRE DURAND lors de *TROUVENR*. Ces noms seraient convertis respectivement dans leurs formes standard DURANT PIERRE et DURAND PIERRE, et l'on ne trouverait aucune correspondance lors de la recherche, bien que l'enregistrement recherché se trouve bien là. Nous n'allons pas essayer de résoudre ce problème, car une solution satisfaisante représenterait une tâche énorme. Pour les lecteurs intéressés par l'expérience, voici quelques indications :

```
DÉBUT recherche de correspondance exacte
  IF la correspondance exacte est trouvée
    THEN PRINT enregistrement complet
    ELSE rechercher dans le tableau une correspondance
         approchée
       IF correspondance approchée trouvée
         THEN PRINT enregistrement correspondant
         ELSE PRINT « AUCUN ENREGISTREMENT
              TROUVÉ »
       FINSI
    FINSI
FINSI
```

La procédure de correspondance approchée pourrait suivre le déroulement suivant :

```
DÉBUT correspondance approchée
  Tableau de recherche de correspondance exacte des noms
     de famille
  IF correspondance exacte du nom de famille
    THEN rechercher prénoms pour correspondance maximale
    PRINT enregistrement pour correspondance maximale
    ELSE rechercher nom de famille pour correspondance
       maximale
      IF nom de famille correspondant trouvé
        THEN PRINT enregistrement pour correspondance
           maximale
      FINSI
    FINSI
FINSI
```

La procédure de correspondance maximale pourrait être définie en gros comme le fait de trouver la chaîne cible avec le nombre maximal de caractères communs avec ceux de la chaîne clé. Elle pourrait accepter une situation dans laquelle la chaîne clé serait entièrement contenue dans la chaîne cible, ou vice versa. Il n'y a pas de solutions simples, mais de nombreuses façons d'entreprendre la programmation.

Le fragment de programme proposé présente un « effet secondaire ». Supposons que la séquence d'événements suivante ait lieu. Il y a dix enregistrements dans le fichier de données. Faites tourner le programme, puis utilisez *AJOUTENR* pour ajouter un nouvel enregistrement, suivi de *TROUVENR* pour localiser un enregistrement. Enfin, en faisant tourner *QUITTE* pour sauvegarder le fichier et terminer le programme, l'enregistrement rajouté ne sera pas sauvegardé (quoique tous les autres le soient). Cela résulte directement de quelque chose qui s'est passé dans l'exécution de *TROUVENR*. Pouvez-vous deviner pourquoi l'enregistrement ajouté ne sera pas sauvegardé?

Prochainement, nous expliquerons comment éviter cette perte de données, nous montrerons pourquoi on se sert de la variable ACT et décrirons comment effacer ou modifier un enregistrement. D'autres options du menu principal (*TRVILLE*, etc.) sont très semblables aux routines que nous avons déjà expliquées. Le lecteur les mettra en œuvre lui-même s'il en a besoin.

Pour finir, considérons ce qui se passerait s'il y avait exactement 50 enregistrements dans le fichier de données et si la routine *TROUVENR* modifiée (qui appelle *MODNOM*) était utilisée. (Indication : TAILLE aura la valeur 51.)

Variantes de basic

SPECTRUM

Pour le Spectrum, les modifications suivantes sont nécessaires :

```
13000 REM VERSION DE *TROUVENR* POUR TEST
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT « CLÉ DE RECHERCHE »; C$

13100 LET INF = 1
13110 LET SUP = TAILLE - 1
13120 FOR L = 1 TO 1
13130 LET MED = INT((INF + SUP)/2)
13140 IF M$(MED) < >C$ THEN LET L = 0
13150 IF M$(MED) < C$ THEN LET INF = MED + 1
13160 IF M$(MED) >C$ THEN LET SUP = MED - 1
13170 IF INF >SUP THEN LET L = 1
13180 NEXT L
:
13200 IF INF >SUP THEN PRINT
      « ENREGISTREMENT NON TROUVÉ »
13210 IF INF < = SUP THEN PRINT
      « L'ENREGISTREMENT EST LE NO »; MED
:
13240 STOP
13250 RETURN
```

Remarquer cette fois encore le problème des variables de chaîne à une seule lettre sur le Spectrum : ici C$ a été substitué à CLERECH$.

```
13000 REM VERSION DE *TROUVENR* POUR TEST
13010 IF RMOD = 1 THEN GOSUB 11200
13020 INPUT « CLÉ DE RECHERCHE »; CLERECH$
13030 REM
13040 REM
13050 REM
13060 REM
13070 REM
13080 REM
13090 REM
13100 LET INF = 1
13110 LET SUP = TAILLE - 1
13120 FOR L = 1 TO 1
13130 LET MED = INT((INF + SUP)/2)
13140 IF MODCHP$(MED) < >CLERECH$ THEN L = 0
13150 IF MODCHP$(MED) < CLERECH$ THEN INF = MED + 1
13160 IF MODCHP$(MED) >CLERECH$ THEN SUP = MED - 1
13170 IF INF >SUP THEN L = 1
13180 NEXT L
13190 REM
13200 IF INF >SUP THEN PRINT « ENREGISTREMENT NON TROUVÉ »
13210 IF INF < = SUP THEN PRINT « L'ENREGISTREMENT EST LE NO »; MED
13220 REM
13230 REM
13240 STOP
13250 RETURN
```

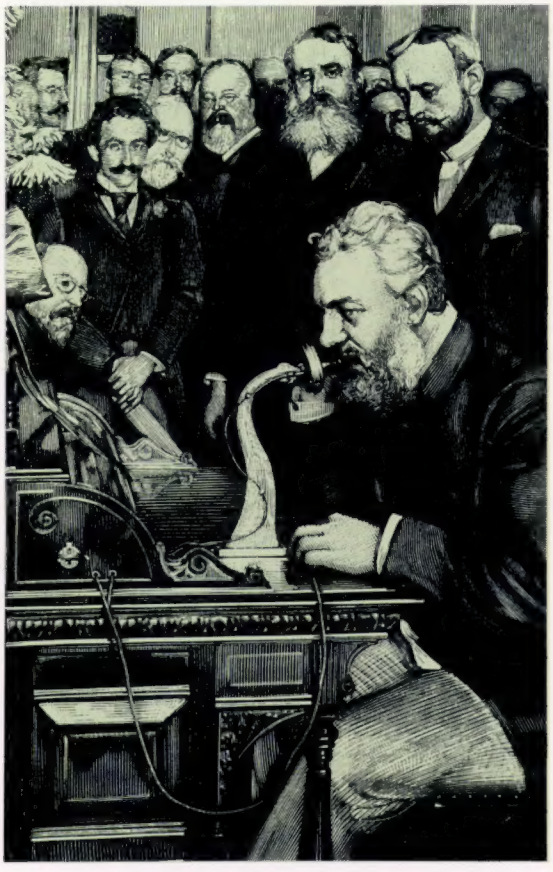
Tony Lodge

Ma Bell

On doit aux laboratoires Bell bien des découvertes dans l'informatique — dans le domaine des matériels comme dans celui des logiciels.

Il y a une centaine d'années, la reine Victoria s'amusa grandement d'une nouvelle invention qui, de l'île de Wight, lui avait permis de parler à ses ministres restés à Londres. Depuis ces temps héroïques où il fallait actionner la manivelle, le téléphone a bénéficié des multiples progrés nés de la recherche, et dont l'ordinateur est une des principales retombées. Un facteur important à l'origine de ces innovations fut la décision de l'American Telephone and Telegraph Company de créer un organisme de recherche; c'est ainsi qu'en 1925 les laboratoires Bell (connus sous l'expression « Ma Bell ») furent établis à Murray Hill, dans le New Jersey (É.-U.).

Il s'agit d'une institution très particulière : elle se consacre uniquement à la recherche fondamentale. ATT est pourtant une compagnie aux buts uniquement commerciaux, mais qui considère que c'est là un investissement à long terme, qui pourra se révéler profitable. Les scientifiques de valeur qui y travaillent sont tenus à l'écart des problèmes techniques quotidiens, et poursuivent leurs travaux comme ils l'entendent. Au fil des années cette politique a valu aux laboratoires Bell deux prix Nobel, et de nombreuses découvertes dans des domaines très différents. Nous n'évoquerons ici que celles qui ont un rapport direct avec la création de l'ordinateur.



Premier coup de fil
Les laboratoires Bell doivent leur nom à Alexander Graham Bell (1847-1922), en qui on s'accorde à voir l'inventeur du téléphone en 1876. D'après la tradition, il est admis que les premiers mots jamais transmis le long de fils électriques furent un appel de Bell à son assistant, situé dans la pièce à côté : « Venez ici, M. Watson, j'ai besoin de vous ! »

Dans les années trente, le téléphone se faisait toujours plus complexe et automatisé. Les messages étaient envoyés sous forme analogique le long des fils, et les appels parvenaient à destination en utilisant l'information contenue dans un code numérique intégré dans un cadran. Le numéro ainsi composé était d'abord traduit : un signal analogique était transformé en une série d'impulsions numériques. Elles étaient maintenues en mémoire grâce à des commutateurs-relais, puis la liaison était assurée par un ensemble de commutateurs à barres croisées, qui comptaient les impulsions et les transformaient en coordonnées sur un panneau de distribution électromécanique. Tous les éléments de l'ordinateur étaient là : il ne manquait plus que quelqu'un pour s'en rendre compte.

Un mathématicien qui travaillait aux laboratoires Bell, George Stibitz, nota la similitude entre le décompte des impulsions et leur addition ultérieure. Travaillant chez lui sur une simple table de cuisine et utilisant de vieux commutateurs à barres croisées et des relais électromécaniques, il construisit les premiers circuits d'ordinateur.

Il entreprit alors des recherches avec un ingénieur nommé Samuel B. Williams, qui construisait des circuits à commutateurs depuis vingt-cinq ans. Tous deux mirent au point un calculateur de nombres complexes. (Les nombres complexes font intervenir des nombres dits « imaginaires » — les racines carrées des nombres négatifs — et ils sont indispensables à la résolution complète des équations polynomiales.) Leurs travaux commencèrent en 1937 et entraînèrent l'utilisation de 450 relais et de 10 commutateurs à barres croisées. L'appareil opérait en notation binaire et pouvait diviser deux nombres de huit chiffres en trente secondes. Il devint opérationnel le 8 janvier 1940 et, en septembre de la même année, il fut présenté devant l'American Mathematical Society à l'université de Dartmouth (où plus tard naîtra le BASIC). Des claviers de machines à écrire reliés par téléphone au calculateur, installé à New York, permettaient un accès multiple à distance... le traitement décentralisé !

On doit aux laboratoires Bell quelques innovations mineures (certains dispositifs pour les têtes de lecture des bandes magnétiques, les amplificateurs à feedback négatif); mais leur principal titre de gloire reste l'invention du transistor, mis au point en 1947 par Bardeen, Brattain et Shockley. Cette invention a rendu possible l'apparition des ordinateurs de deuxième génération.

PROGRAMME N° 9

CROCODILES

LES SOUS-PROGRAMMES
Utilisation de GOSUB-RETURN

Des images sur l'écran! Pourquoi pas. Commençons par dessiner un « crocodile » bleu à tête blanche et à pattes orange.
Voici le programme vous permettant de dessiner en GR.

```
]LIST

10  HOME
20  REM GENERATION D'UN CROCO ALEATOIRE
30  GR
40  COLOR = 7: REM BLEU CLAIR
50  PLOT 15,15
60  HLIN 15,18 AT 16
70  COLOR = 9: REM ORANGE
80  PLOT 15,17
90  PLOT 17,17
100 COLOR = 15: REM BLANC
110 PLOT 14,15
```

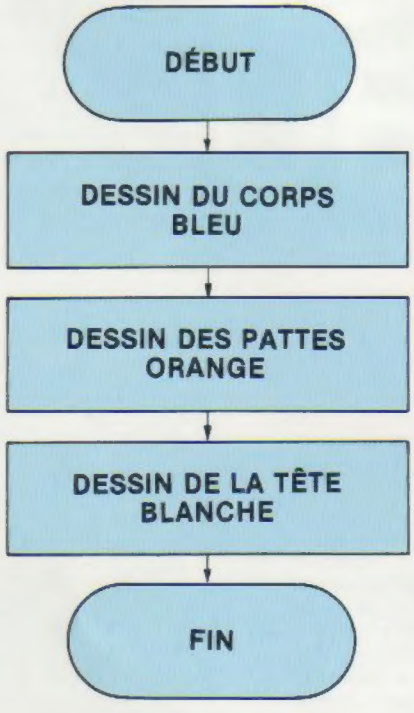
10 efface l'écran
30 GRAPHIQUE

40
50 dessin en bleu
60

70
80 dessin en orange
90

100
110 dessin en blanc

ORGANIGRAMME

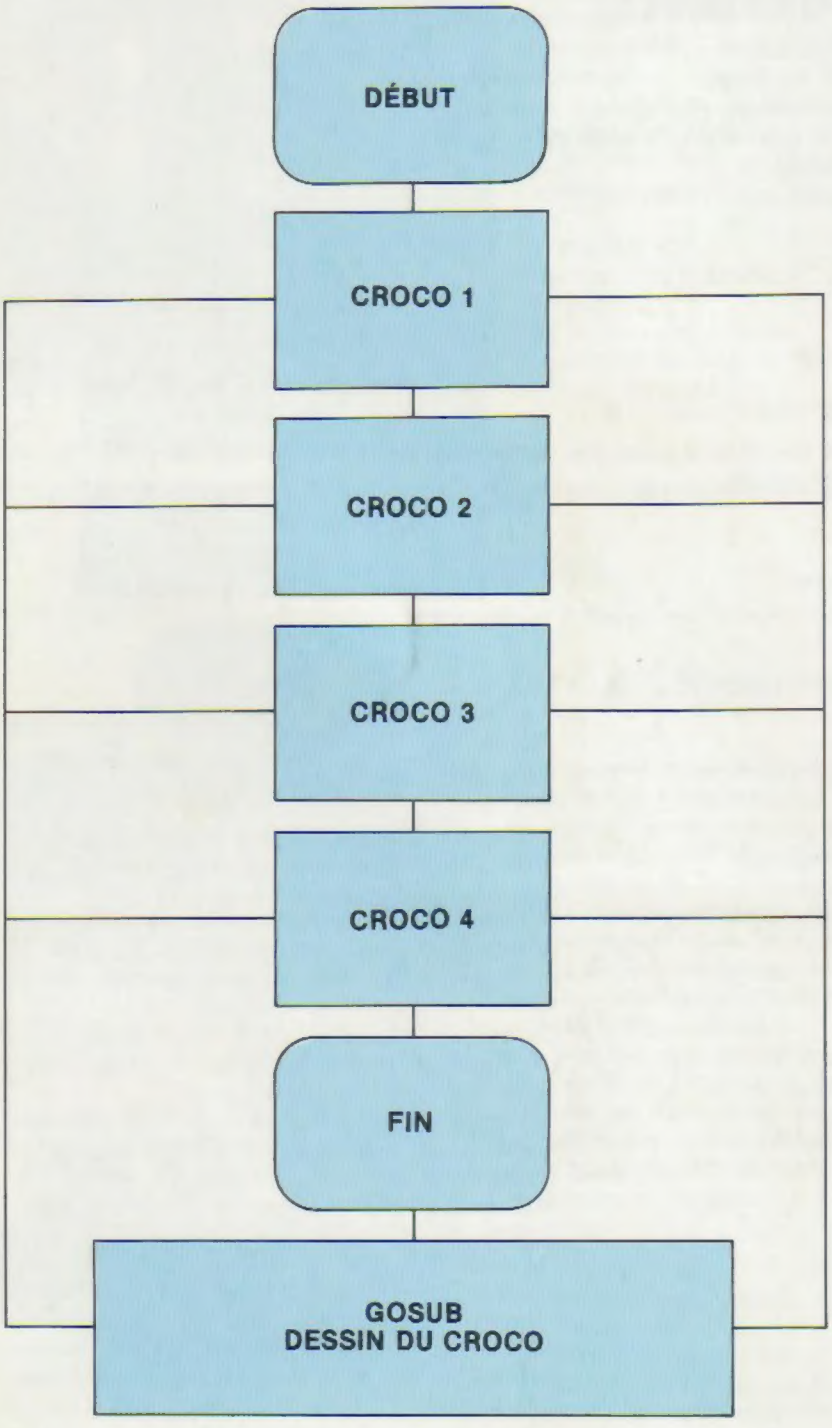


Ce programme est correct; il génère un crocodile bleu, à tête blanche et pattes orange.
Supposons que vous vouliez dessiner un second crocodile identique à un autre endroit de l'écran. Vous pourriez réécrire le programme en y introduisant de nouvelles valeurs de X et Y, ce serait pénible et un peu inutile. En effet, il existe un moyen d'utiliser ce programme sans le réécrire complètement à chaque fois.
On peut, à partir d'un point P de coordonnées (A, B), déplacer ce point P vers la droite en augmentant la valeur de l'abcisse A.
Exemple : pour déplacer P (2,5) de 15 colonnes vers la droite, il faut ajouter 15 à 2. Le point P prend les coordonnées (17,5). De même, on déplace le point vers le bas; en diminuant B, on déplace P vers le haut de l'écran.
On peut aussi réécrire le programme et centrer le crocodile en un point presque quelconque de l'écran. Presque quelconque, car, si on choisit un point de centrage au bord de l'écran, le crocodile disparaît, et la machine écrit un message d'erreur.

Voici le programme amélioré :

```
10 REM GENERATION D'UN CROCO ALEATOIRE
20 COLOR = 7
30 PLOT X - 1,Y - 1
40 HLIN X - 1,X + 2 AT Y
50 COLOR = 9
60 PLOT X - 1,Y + 1
70 PLOT X + 1,Y + 1
80 COLOR = 15
90 PLOT X - 2,Y - 1
```

ORGANIGRAMME



Vous remarquerez qu'il n'y a pas d'instruction GR car nous voulons employer cette partie du programme pour placer plusieurs crocodiles sur l'écran.
L'instruction GR aurait effacé le dessin précédent avant d'autoriser l'affichage d'un nouveau crocodile.
Il est donc impossible d'exécuter ce programme sous cette forme. Il faut tout d'abord passer en mode GR et donner la couleur de X et Y.

On peut écrire par exemple :

```
$LIST

5 GR
6 X = 32
7 Y = 35
```

En tapant RUN, on obtient le dessin d'un crocodile à l'endroit désiré, mais le programme stoppe aussitôt.
Pour dessiner plusieurs crocodiles : quatre, par exemple,

Tapez

```
$LIST

1   GR
2   X = 32:Y = 20
3   GOSUB 10
4   X = 2:Y = 18
5   GOSUB 10
6   X = 36:Y = 35
7   GOSUB 10
8   X = 4:Y = 5: GOSUB 10
9   END
10  REM GENERATION D'UN CROCO ALEATOIRE
20  COLOR = 7
30  PLOT X - 1,Y - 1
40  HLIN X - 1,X + 2 AT Y
50  COLOR = 9
60  PLOT X - 1,Y + 1
70  PLOT X + 1,Y + 1
80  COLOR = 15
90  PLOT X - 2,Y ≠ 1
100 RETURN
```

Vous constatez l'utilisation d'une nouvelle instruction GOSUB 10.
GOSUB 10 commande à la machine d'aller (GO) au sous-programme (SUB routine) débutant en 10 et de commencer l'exécution à cette instruction précise. Elle commande également à la machine de revenir à la ligne suivant GOSUB 10 lorsque le déroulement du sous-programme est terminé.
La machine reconnaît la fin du sous-programme à l'instruction RETURN. Ici RETURN est en ligne 100.